Anonymous, Robust Post-Quantum Public Key Encryption

Varun Maram
Applied Cryptography Group
ETH Zurich

APPLIED CRYPTO GROUP

Joint work with Paul Grubbs and Kenneth G. Paterson

[Full version of paper: https://eprint.iacr.org/2021/708.pdf]

NIST PQC Round-3 KEMs

PQC Standardization Process: Third Round Candidate Announcement

NIST is announcing the third round finalists of the NIST Post-Quantum Cryptography Standardization Process. More details are included in NISTIR 8309.

July 22, 2020

It has been almost a year and a half since the second round of the NIST PQC Standardization Process began. After careful consideration, NIST would like to announce the candidates that will be moving on to the third round.

Third Round Finalists	Alternate Candidates
Public-Key Encryption/KEMs	Public-Key Encryption/KEMs
Classic McEliece	BIKE
CRYSTALS-KYBER	FrodoKEM
NTRU	HQC
SABER	NTRU Prime
	SIKE

ORGANIZATIONS

Information Technology Laboratory

Computer Security Division

Cryptographic Technology Group

NIST PQC Round-3 KEMs

PQC Standardization Process: Third Round Candidate Announcement

NIST is announcing the third round finalists of the NIST Post-Quantum Cryptography Standardization Process. More details are included in NISTIR 8309.

July 22, 2020

It has been almost a year and a half since the second round of the NIST PQC Standardization Process began. After careful consideration, NIST would like to announce the candidates that will be moving on to the third round.

Third Round Finalists	**Alternate Candidates**
Public-Key Encryption/KEMs	Public-Key Encryption/KEMs
Classic McEliece	BIKE
CRYSTALS-KYBER	FrodoKEM
NTRU	HQC
SABER	NTRU Prime
	SIKE

ORGANIZATIONS

Information Technology Laboratory

Computer Security Division

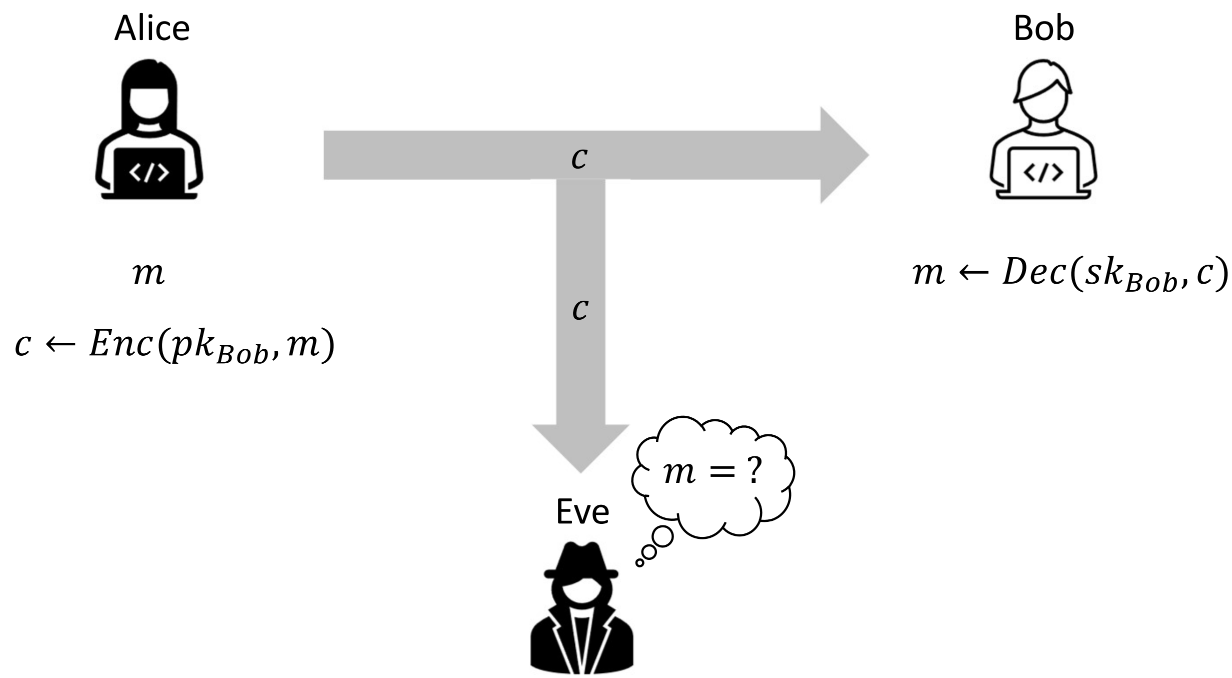
Cryptographic Technology Group

4.A.2 Security Definition for Encryption/Key-Establishment

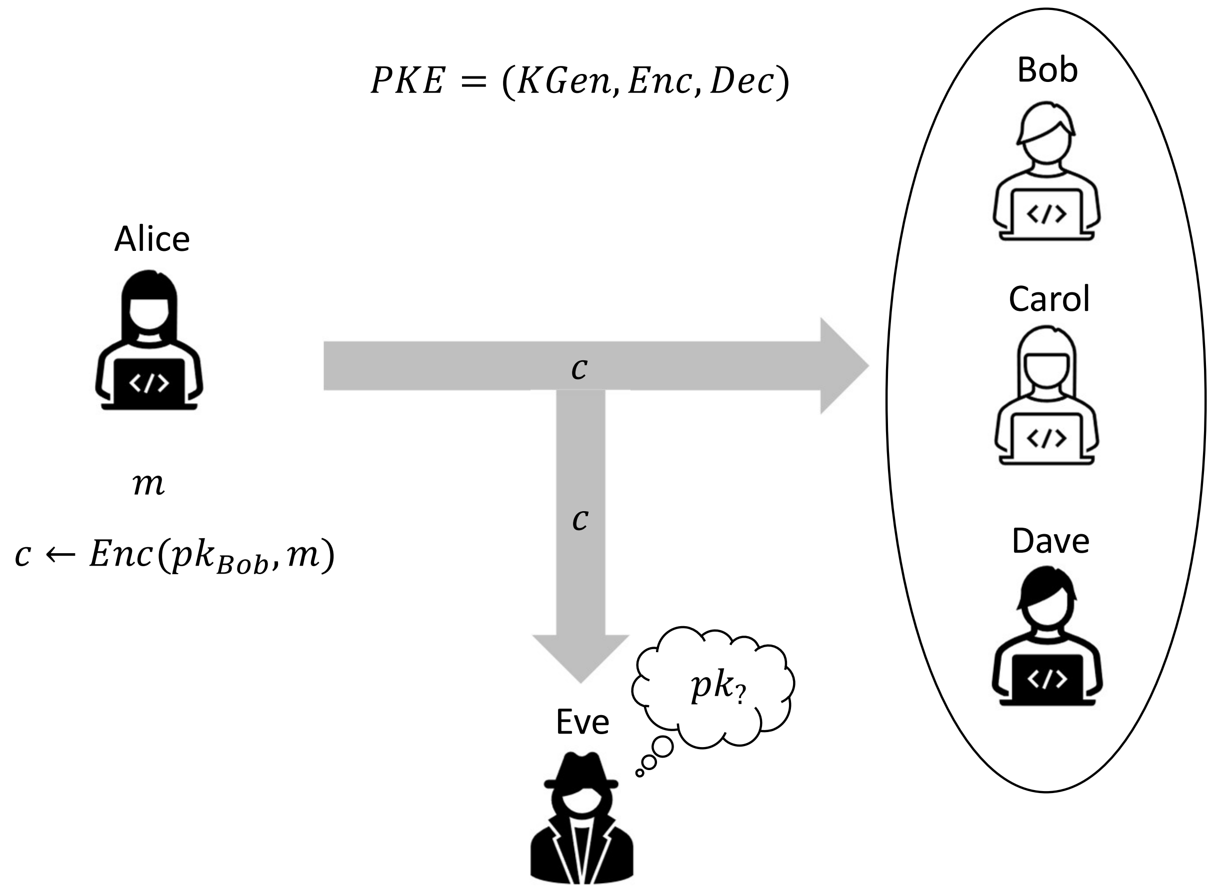
NIST intends to standardize one or more schemes that enable "semantically secure" encryption or key encapsulation with respect to adaptive chosen ciphertext attack, for general use. This property is generally denoted *IND-CCA2 security* in academic literature.

IND-CCA Security

$$PKE = (KGen, Enc, Dec)$$

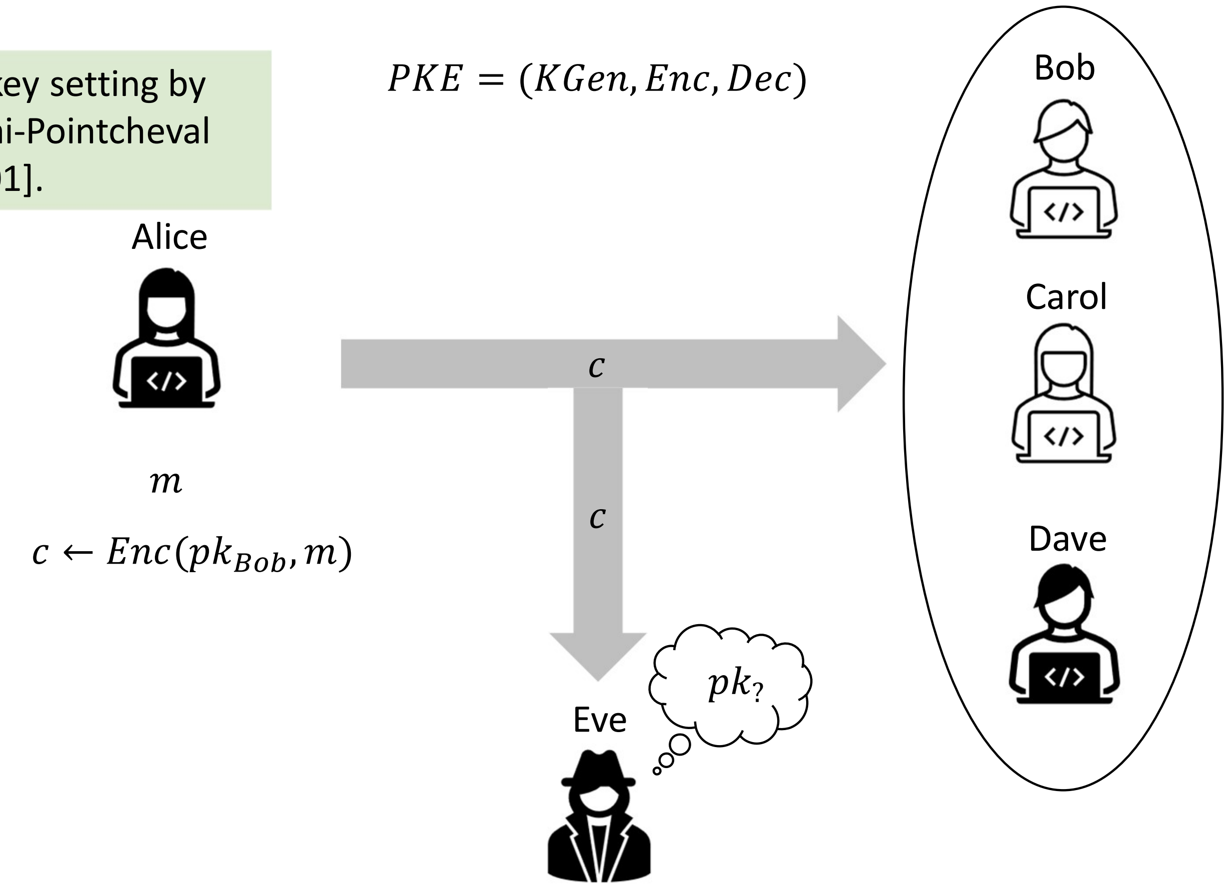


Anonymity (ANO-CCA security)



Anonymity (ANO-CCA security)

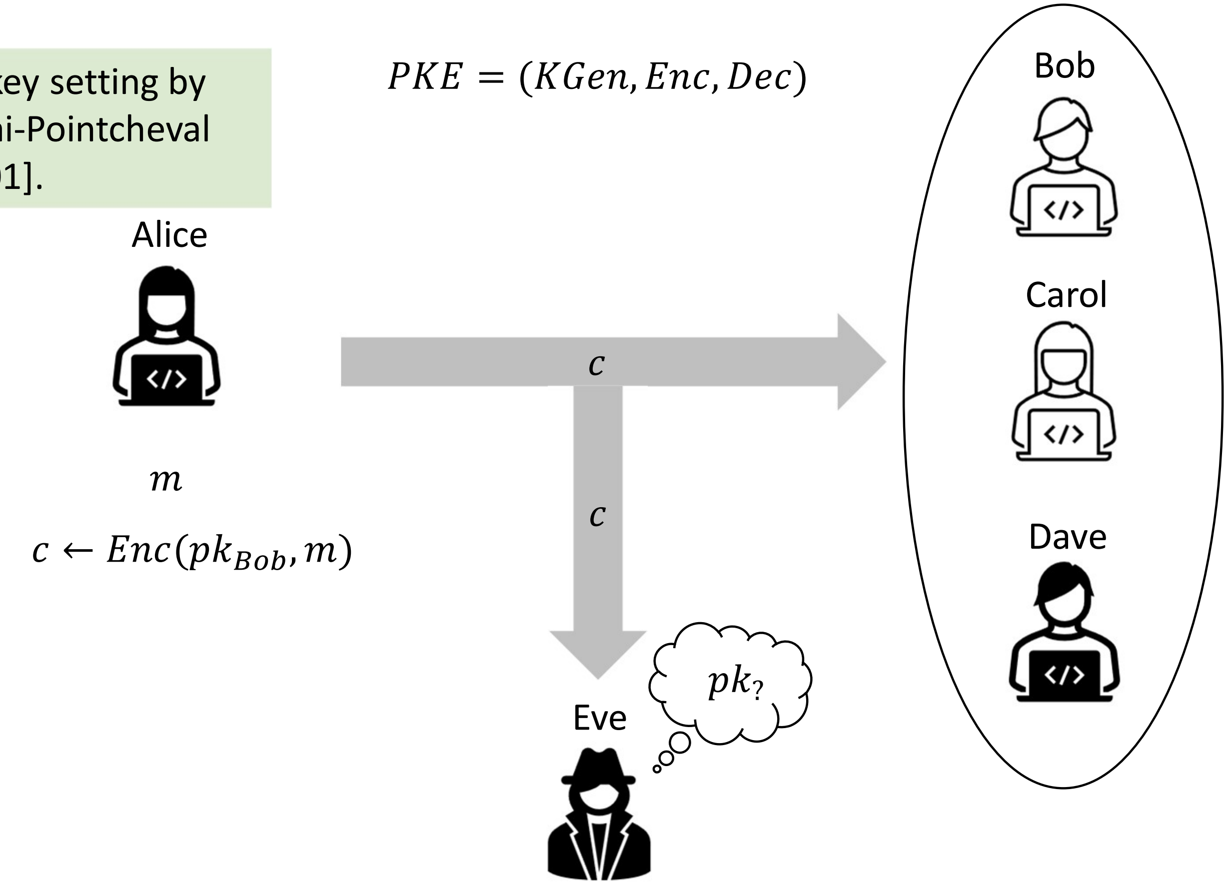
Formalized in a public-key setting by [Bellare-Boldyreva-Desai-Pointcheval @Asiacrypt'01].



Anonymity (ANO-CCA security)

Formalized in a public-key setting by [Bellare-Boldyreva-Desai-Pointcheval @Asiacrypt'01].

$PKE = (KGen, Enc, Dec)$

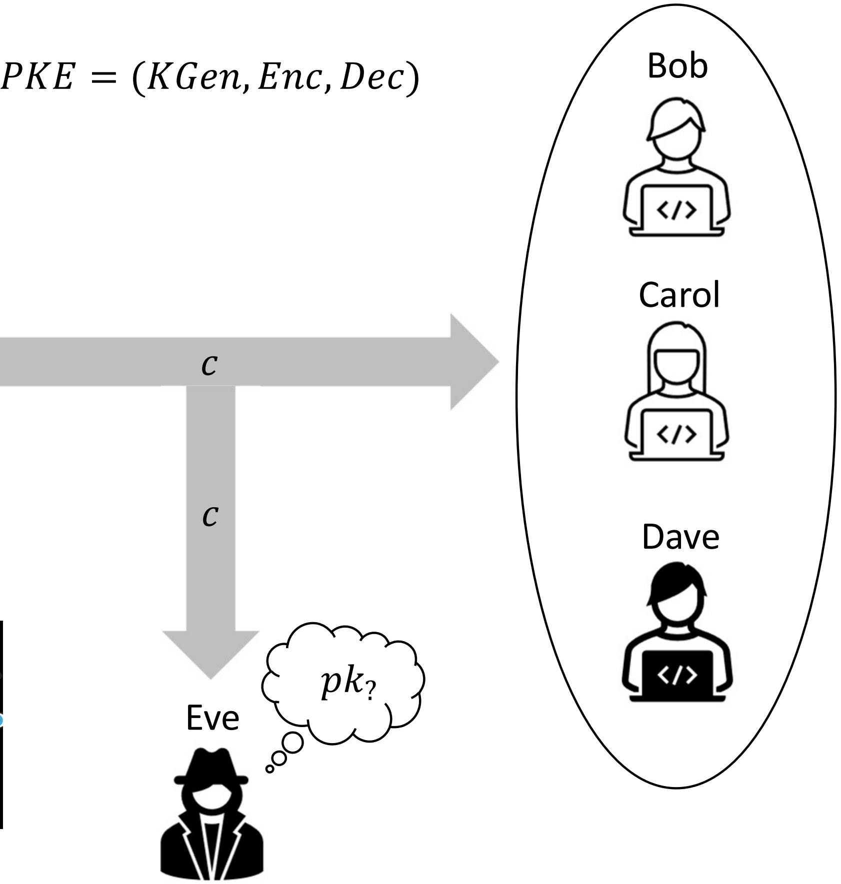


Anonymity (ANO-CCA security)

Formalized in a public-key setting by [Bellare-Boldyreva-Desai-Pointcheval @Asiacrypt'01].

$PKE = (KGen, Enc, Dec)$

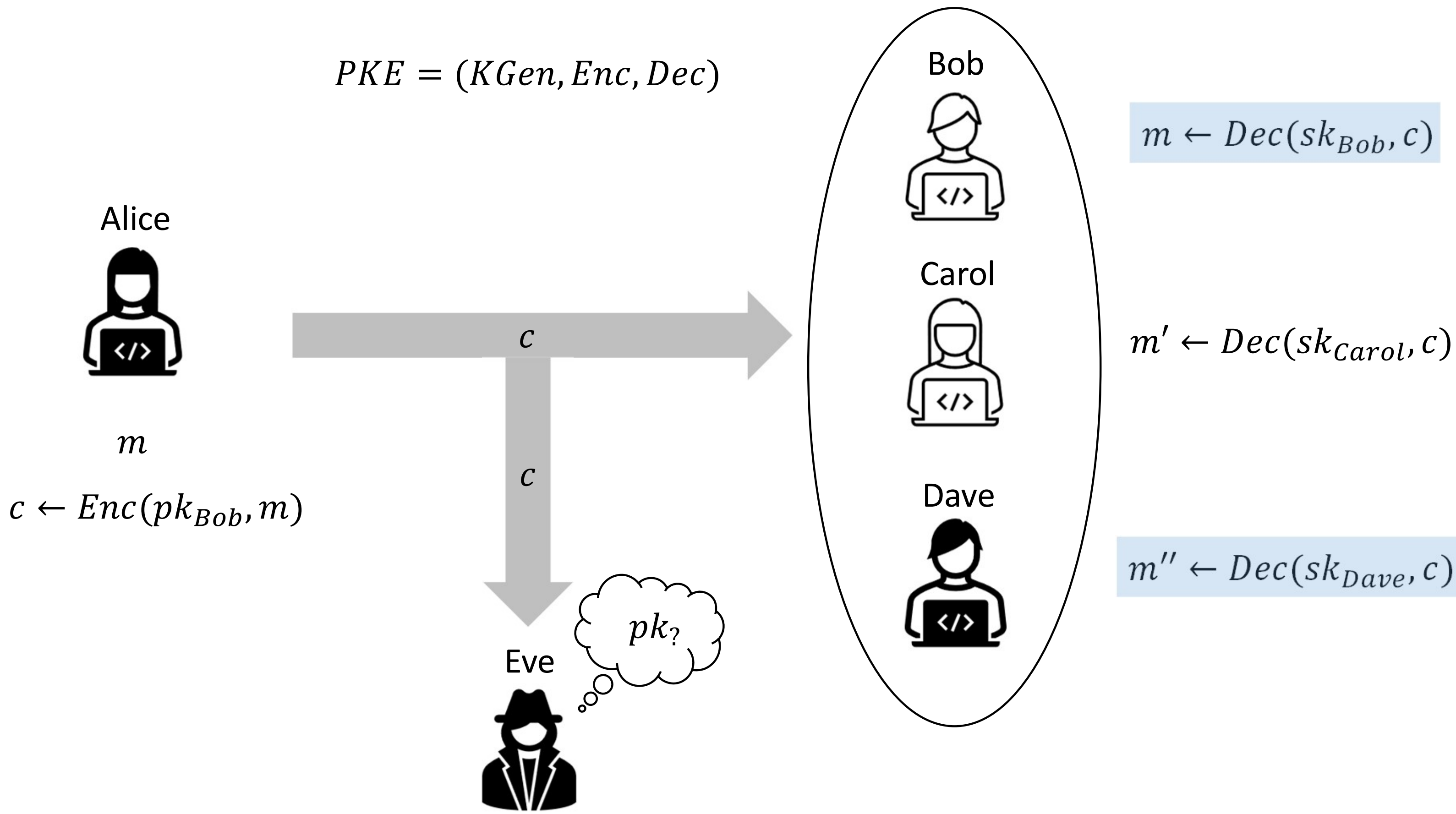
Alice



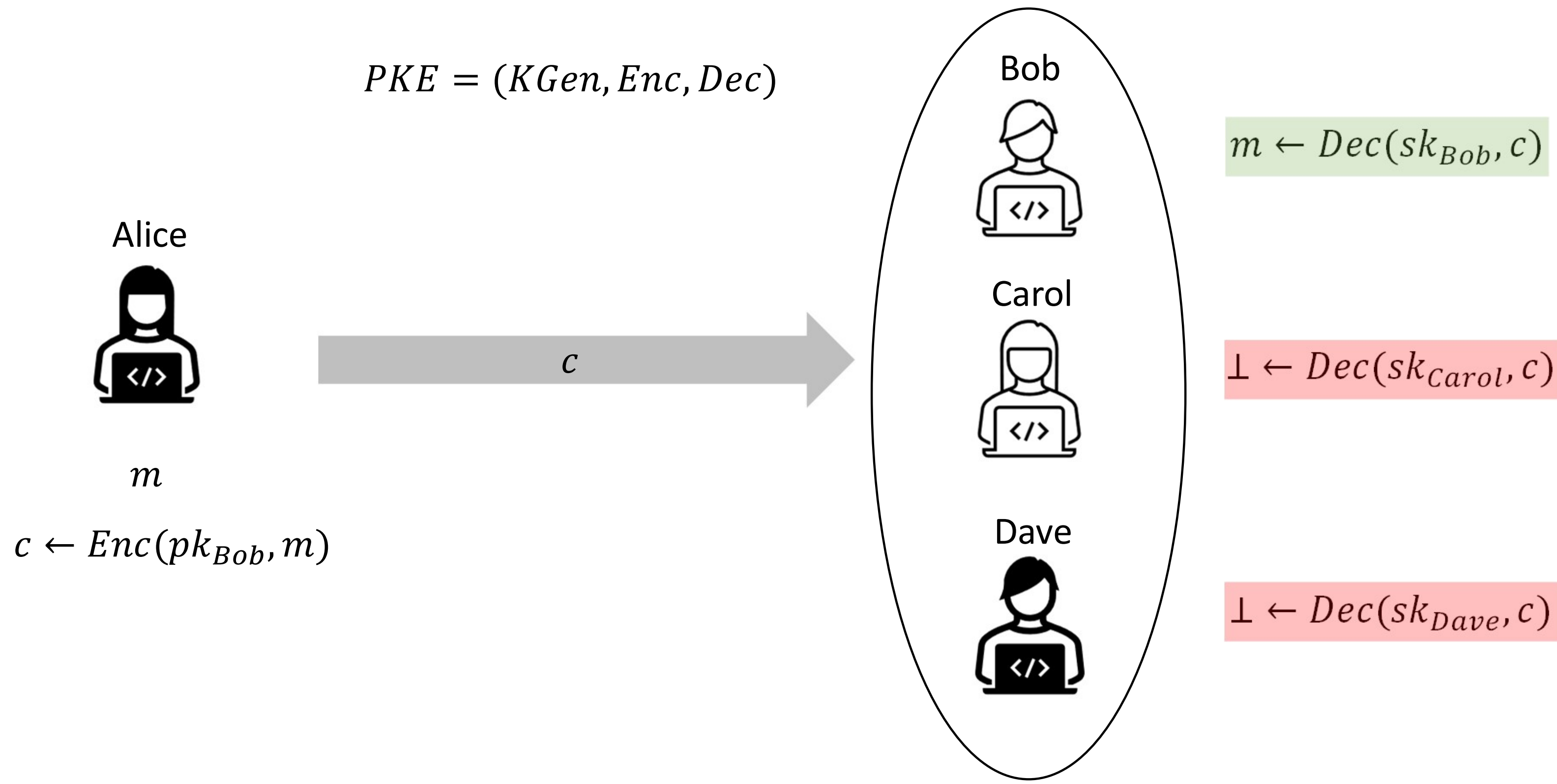
m

$c \leftarrow Enc(pk_{Bob}, m)$

Anonymity (ANO-CCA security)



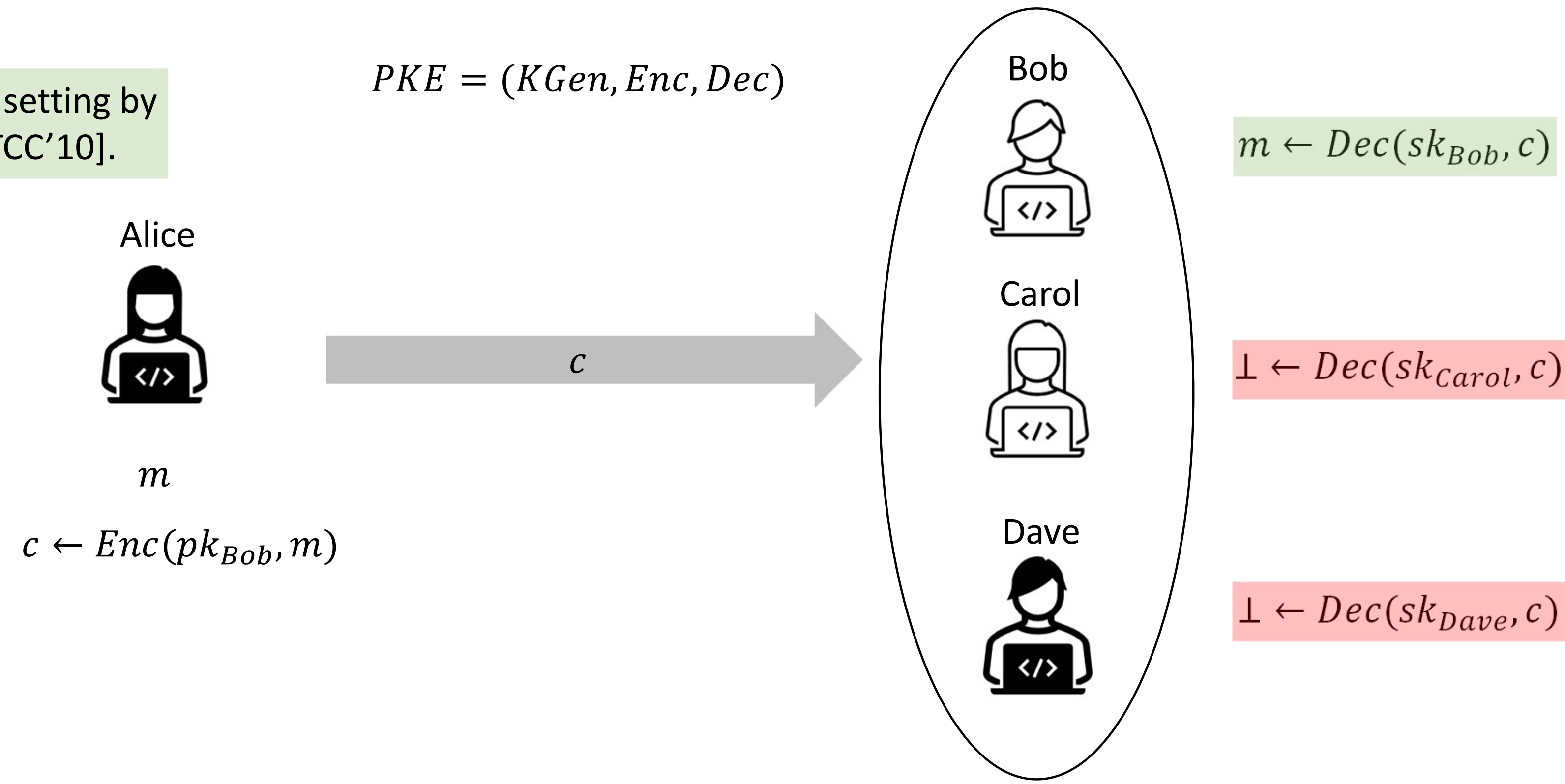
Robustness (SROB-CCA security)



Robustness (SROB-CCA security)

Formalized in a public-key setting by [Abdalla-Bellare-Neven@TCC'10].

$PKE = (KGen, Enc, Dec)$



KEM-DEM Paradigm

Public-Key Encryption/KEMs

Classic McEliece

CRYSTALS-KYBER

NTRU

SABER

Public-Key Encryption/KEMs

BIKE

FrodoKEM

HQC

NTRU Prime

SIKE

KEM-DEM Paradigm

<u>Public-Key Encryption/KEMs</u>

Classic McEliece

CRYSTALS-KYBER

NTRU

SABER

<u>Public-Key Encryption/KEMs</u>

BIKE

FrodoKEM

HQC

NTRU Prime

SIKE

$$PKE = (KGen, Enc, Dec)$$

IND-CCA secure

KEM-DEM Paradigm

Public-Key Encryption/KEMs

Classic McEliece

CRYSTALS-KYBER

NTRU

SABER

$KEM = (KGen, Encap, Decap)$

KEM

IND-CCA secure

Public-Key Encryption/KEMs

BIKE

FrodoKEM

HQC

NTRU Prime

SIKE

$PKE = (KGen, Enc, Dec)$

PKE

IND-CCA secure

KEM-DEM Paradigm

Public-Key Encryption/KEMs

Classic McEliece

CRYSTALS-KYBER

NTRU

SABER

Public-Key Encryption/KEMs

BIKE

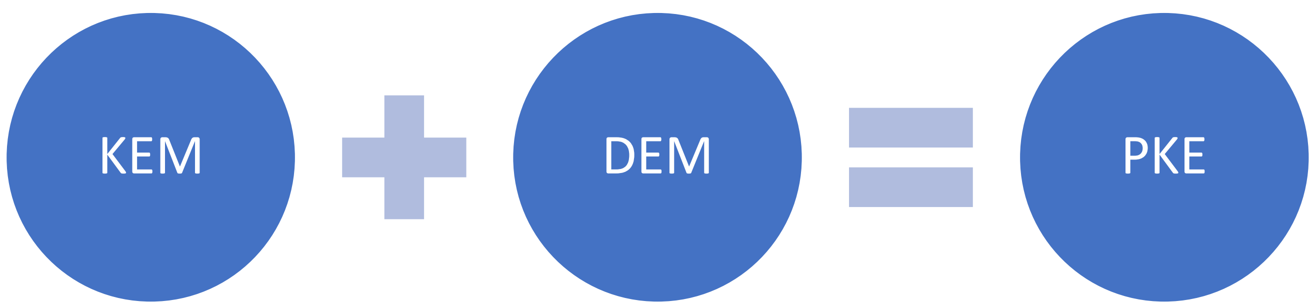
FrodoKEM

HQC

NTRU Prime

SIKE

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



IND-CCA secure

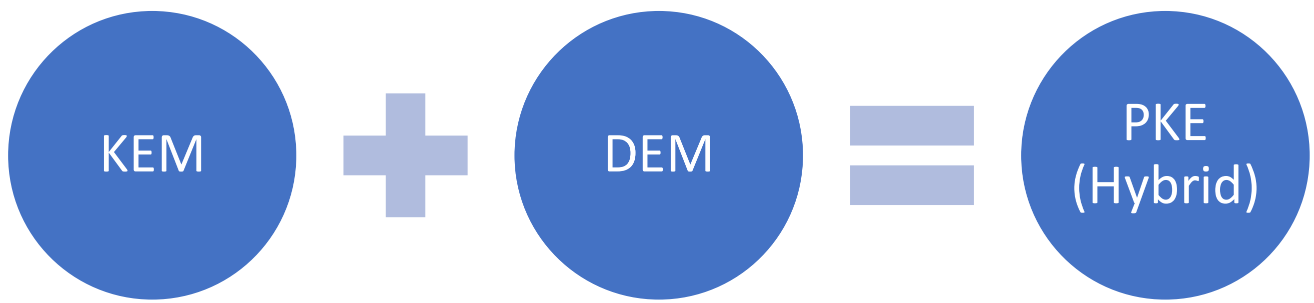
(one-time) authenticated encryption

IND-CCA secure

KEM-DEM Paradigm

Public-Key Encryption/KEMs	Public-Key Encryption/KEMs
Classic McEliece	BIKE
CRYSTALS-KYBER	FrodoKEM
NTRU	HQC
SABER	NTRU Prime
	SIKE

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA secure

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure

KEM-DEM Paradigm

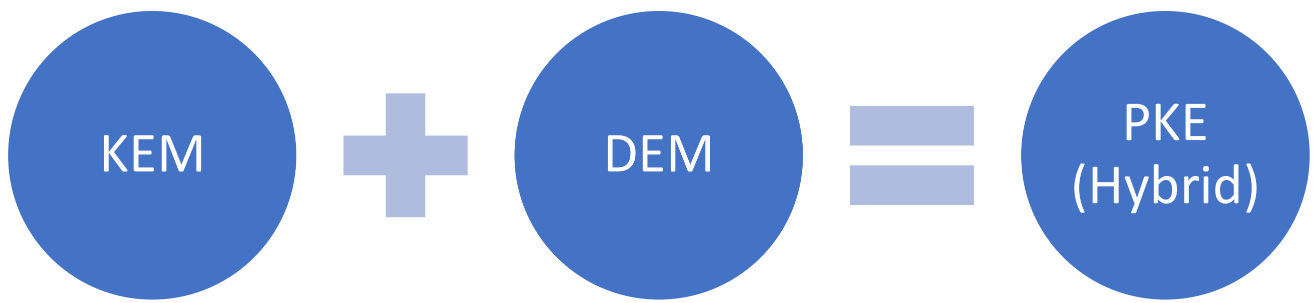
Public-Key Encryption/KEMs

- Classic McEliece
- CRYSTALS-KYBER
- NTRU
- SABER

Public-Key Encryption/KEMs

- BIKE
- FrodoKEM
- HQC
- NTRU Prime
- SIKE

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$ $c_{DEM} \leftarrow Enc^{sym}(k, m)$ $(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

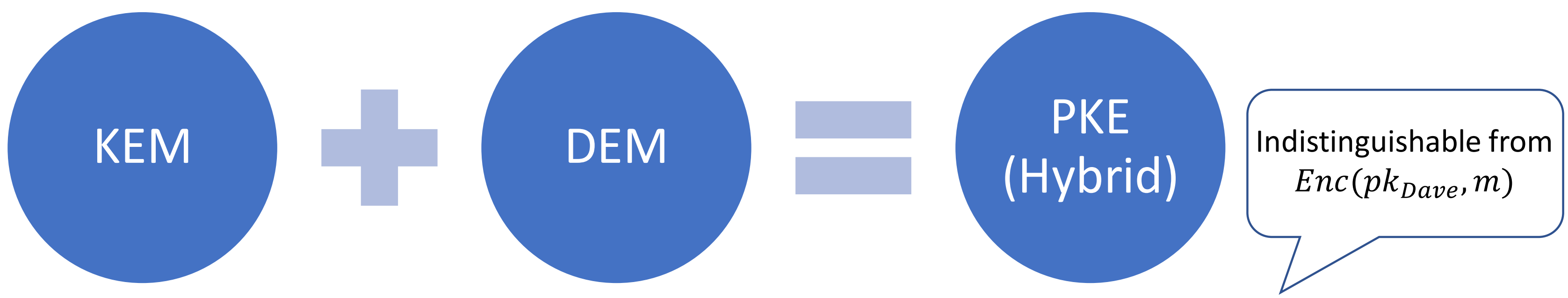
IND-CCA secure +
ANO-CCA secure

KEM-DEM Paradigm

Public-Key Encryption/KEMs
- Classic McEliece
- CRYSTALS-KYBER
- NTRU
- SABER

Public-Key Encryption/KEMs
- BIKE
- FrodoKEM
- HQC
- NTRU Prime
- SIKE

$KEM = (KGen, Encap, Decap)$ $\quad DEM = (Enc^{sym}, Dec^{sym})$ $\quad PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$ $\quad c_{DEM} \leftarrow Enc^{sym}(k, m)$ $\quad (c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure

KEM-DEM Paradigm

Public-Key Encryption/KEMs

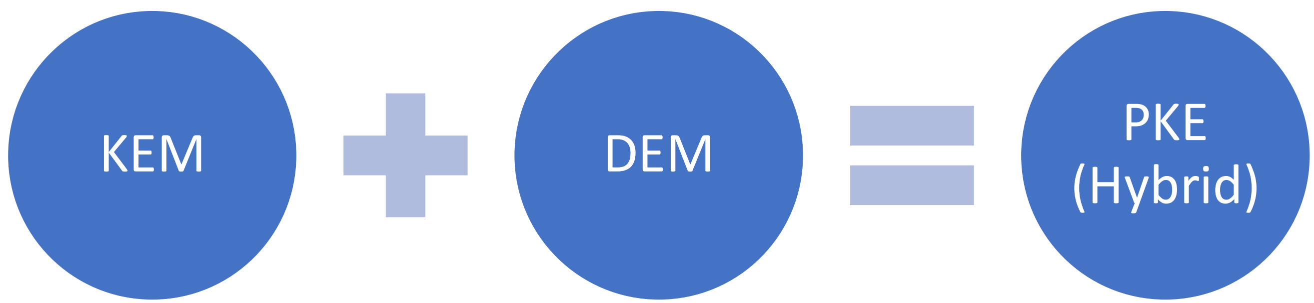
- Classic McEliece
- CRYSTALS-KYBER
- NTRU
- SABER

Public-Key Encryption/KEMs

- BIKE
- FrodoKEM
- HQC
- NTRU Prime
- SIKE

Shown in [Grubbs-Maram-Paterson @Eurocrypt'22]; generalization of [Mohassel@Asiacrypt'10].

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$ $c_{DEM} \leftarrow Enc^{sym}(k, m)$ $(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure

KEM-DEM Paradigm

Public-Key Encryption/KEMs

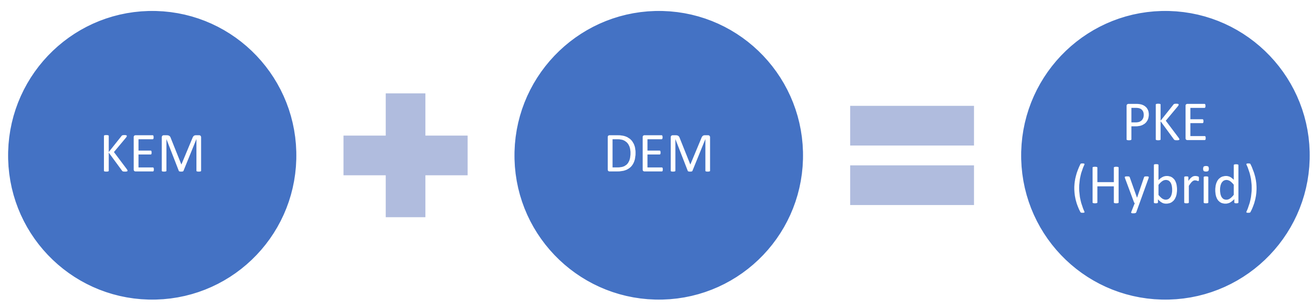
- Classic McEliece
- CRYSTALS-KYBER
- NTRU
- SABER

Public-Key Encryption/KEMs

- BIKE
- FrodoKEM
- HQC
- NTRU Prime
- SIKE

Shown in [Grubbs-Maram-Paterson @Eurocrypt'22]; generalization of [Mohassel@Asiacrypt’10].

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

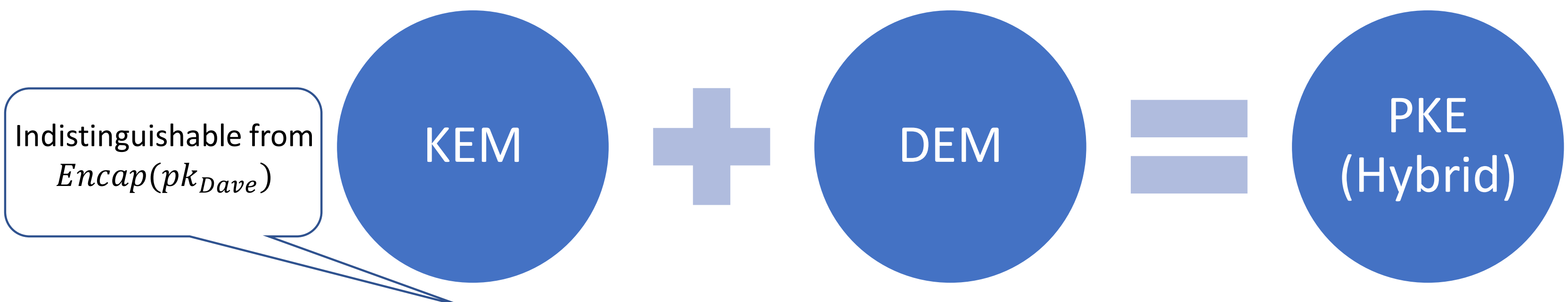
IND-CCA secure + ANO-CCA secure

KEM-DEM Paradigm

Public-Key Encryption/KEMs	Public-Key Encryption/KEMs
Classic McEliece	BIKE
CRYSTALS-KYBER	FrodoKEM
NTRU	HQC
SABER	NTRU Prime
	SIKE

Shown in [Grubbs-Maram-Paterson @Eurocrypt'22]; generalization of [Mohassel@Asiacrypt’10].

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$ $c_{DEM} \leftarrow Enc^{sym}(k, m)$ $(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA + ANO-CCA secure

IND-CCA secure + ANO-CCA secure

KEM-DEM Paradigm

Public-Key Encryption/KEMs

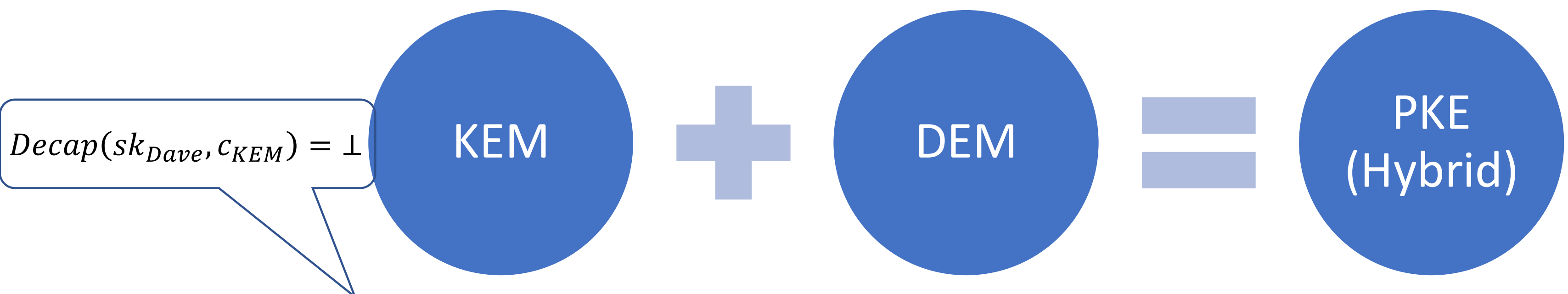
- Classic McEliece
- CRYSTALS-KYBER
- NTRU
- SABER

Public-Key Encryption/KEMs

- BIKE
- FrodoKEM
- HQC
- NTRU Prime
- SIKE

Shown in [Grubbs-Maram-Paterson @Eurocrypt'22]; generalization of [Mohassel@Asiacrypt'10].

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$ $c_{DEM} \leftarrow Enc^{sym}(k, m)$ $(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA + ANO-CCA secure + weakly robust (WROB)

IND-CCA secure + ANO-CCA secure

KEM-DEM Paradigm

Public-Key Encryption/KEMs

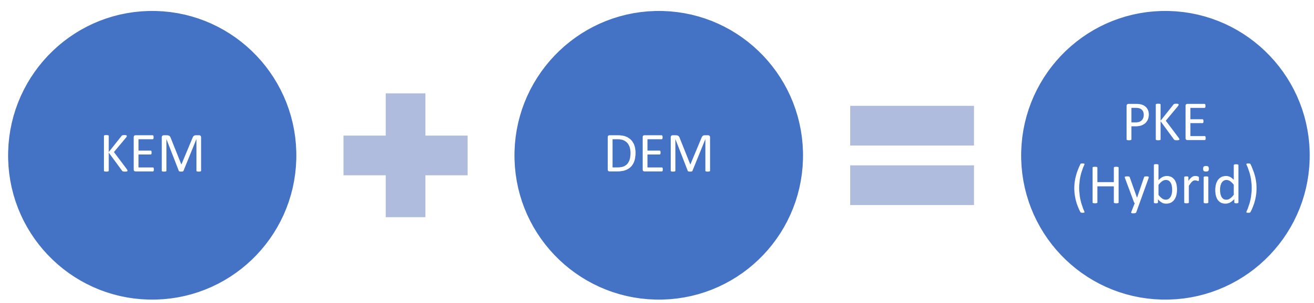
- Classic McEliece
- CRYSTALS-KYBER
- NTRU
- SABER

Public-Key Encryption/KEMs

- BIKE
- FrodoKEM
- HQC
- NTRU Prime
- SIKE

Shown in [Grubbs-Maram-Paterson @Eurocrypt'22]; generalization of [Mohassel@Asiacrypt'10].

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + weakly robust (WROB)

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure ✓

KEM-DEM Paradigm

Public-Key Encryption/KEMs

Classic McEliece

CRYSTALS-KYBER

NTRU

SABER

Public-Key Encryption/KEMs

BIKE

FrodoKEM

HQC

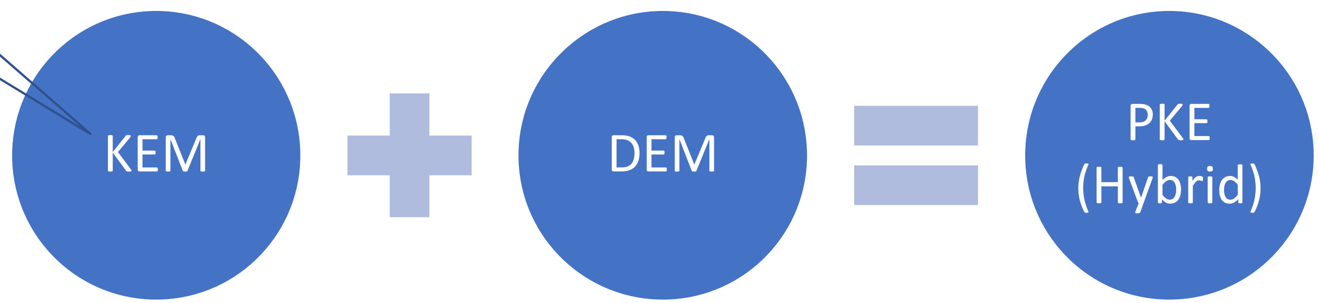
NTRU Prime

SIKE

Shown in [Grubbs-Maram-Paterson @Eurocrypt'22]; generalization of [Mohassel@Asiacrypt'10].

Mohassel only considered KEMs constructed <u>directly</u> from PKE schemes.

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + weakly robust (WROB)

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

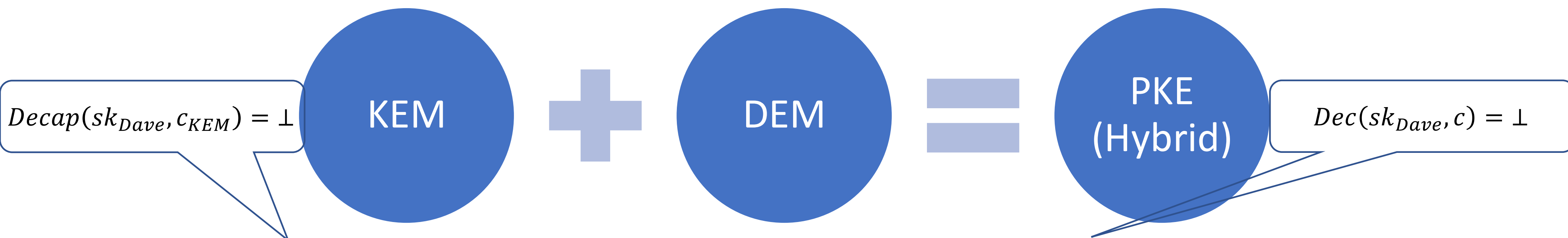
IND-CCA secure + ANO-CCA secure ✓

KEM-DEM Paradigm

Public-Key Encryption/KEMs	Public-Key Encryption/KEMs
Classic McEliece	BIKE
CRYSTALS-KYBER	FrodoKEM
NTRU	HQC
SABER	NTRU Prime
	SIKE

Shown in [Grubbs-Maram-Paterson @Eurocrypt'22]; generalization of [Mohassel@Asiacrypt'10].

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + weakly robust (WROB)

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

$(c) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure + WROB ✓

KEM-DEM Paradigm

Public-Key Encryption/KEMs

- Classic McEliece
- CRYSTALS-KYBER
- NTRU
- SABER

Public-Key Encryption/KEMs

- BIKE
- FrodoKEM
- HQC
- NTRU Prime
- SIKE

Shown in [Grubbs-Maram-Paterson @Eurocrypt'22]; generalization of [Mohassel@Asiacrypt'10].

$KEM = (KGen, Encap, Decap)$ $\quad DEM = (Enc^{sym}, Dec^{sym})$ $\quad PKE = (KGen, Enc, Dec)$

KEM + DEM = PKE (Hybrid)

$Decap(sk_{Bob}, c^*_{KEM}) = \bot$
or,
$Decap(sk_{Dave}, c^*_{KEM}) = \bot$

c^*_{KEM} arbitrary

$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + strongly robust (SROB)

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

$(c) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure + SROB ✓

$Dec(sk_{Bob}, c^*) = \bot$
or,
$Dec(sk_{Dave}, c^*) = \bot$

(c^*) arbitrary

KEM-DEM Paradigm

Public-Key Encryption/KEMs

Classic McEliece

CRYSTALS-KYBER

NTRU

SABER

Public-Key Encryption/KEMs

BIKE

FrodoKEM

HQC

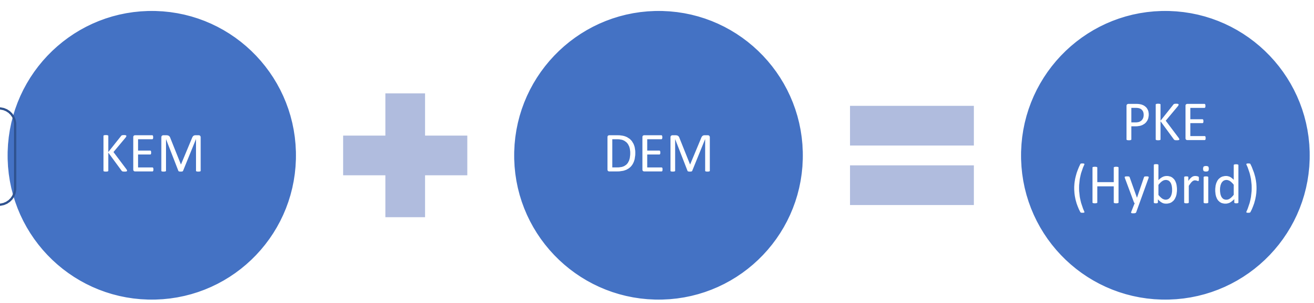
NTRU Prime

SIKE

Shown in [Grubbs-Maram-Paterson @Eurocrypt'22]; generalization of [Mohassel@Asiacrypt'10].

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$

$Decap(sk_{Dave}, c_{KEM}) = \bot$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + weakly robust (WROB)

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure + WROB

KEM-DEM Paradigm

Public-Key Encryption/KEMs	Public-Key Encryption/KEMs
Classic McEliece	BIKE
CRYSTALS-KYBER	FrodoKEM
NTRU	HQC
SABER	NTRU Prime
	SIKE

Shown in [Grubbs-Maram-Paterson @Eurocrypt'22]; generalization of [Mohassel@Asiacrypt'10].

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$

KEM + DEM = PKE (Hybrid)

$Decap(sk_{Dave}, c_{KEM}) = \perp$

... is also necessary.

$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + weakly robust (WROB)

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure + WROB

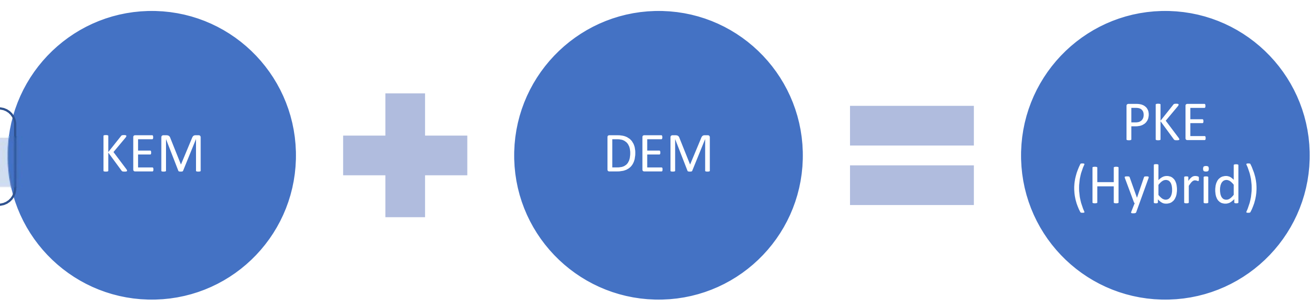
KEM-DEM Paradigm

Public-Key Encryption/KEMs
- Classic McEliece
- CRYSTALS-KYBER
- NTRU
- SABER

Public-Key Encryption/KEMs
- BIKE
- FrodoKEM
- HQC
- NTRU Prime
- SIKE

Shown in [Grubbs-Maram-Paterson @Eurocrypt'22]; generalization of [Mohassel@Asiacrypt'10].

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$Decap(sk_{Dave}, c_{KEM}) = \perp$

... is also necessary.

$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + weakly robust (WROB)

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure + WROB

KEM-DEM Paradigm

Public-Key Encryption/KEMs

- Classic McEliece
- CRYSTALS-KYBER
- NTRU
- SABER

"Implicit-rejection" KEMs!

Public-Key Encryption/KEMs

- BIKE
- FrodoKEM
- HQC
- NTRU Prime
- SIKE

Shown in [Grubbs-Maram-Paterson @Eurocrypt'22]; generalization of [Mohassel@Asiacrypt'10].

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$

KEM + DEM = PKE (Hybrid)

$Decap(sk_{Dave}, c_{KEM}) = \perp$

... is also necessary.

$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + weakly robust (WROB)

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure + WROB

KEM-DEM Paradigm

Public-Key Encryption/KEMs

- Classic McEliece
- CRYSTALS-KYBER
- NTRU
- SABER

"Implicit-rejection" KEMs!

Cannot be even weakly robust.

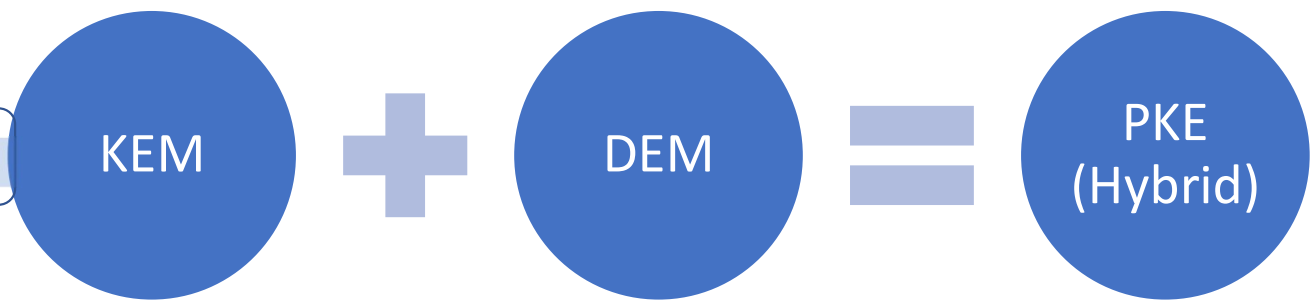
Public-Key Encryption/KEMs

- BIKE
- FrodoKEM
- HQC
- NTRU Prime
- SIKE

Shown in [Grubbs-Maram-Paterson @Eurocrypt'22]; generalization of [Mohassel@Asiacrypt'10].

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$

$Decap(sk_{Dave}, c_{KEM}) = \perp$



... is also necessary.

$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + weakly robust (WROB)

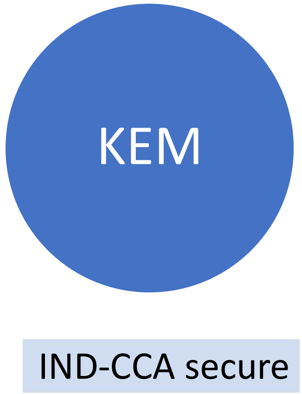
$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

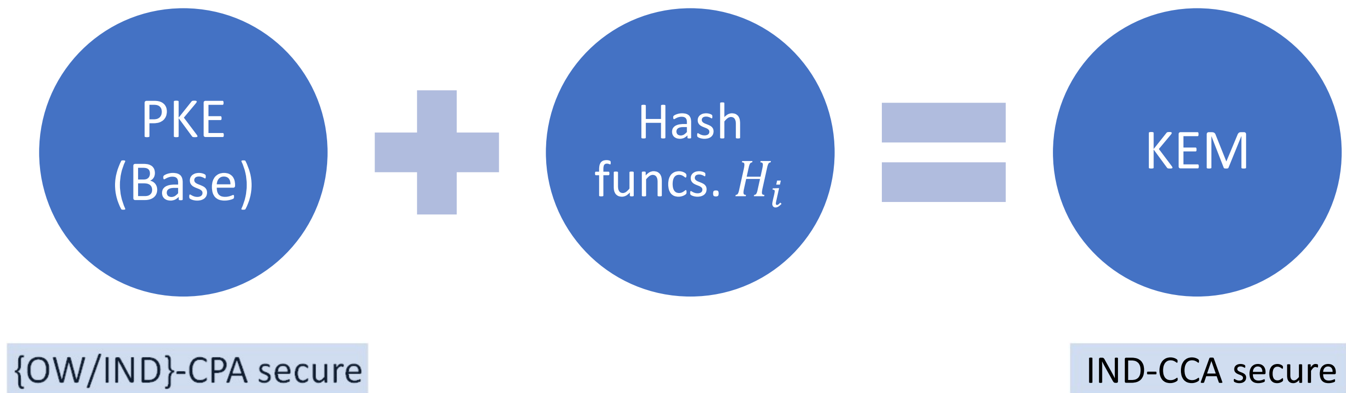
IND-CCA secure + ANO-CCA secure + WROB

Fujisaki-Okamoto Transformation

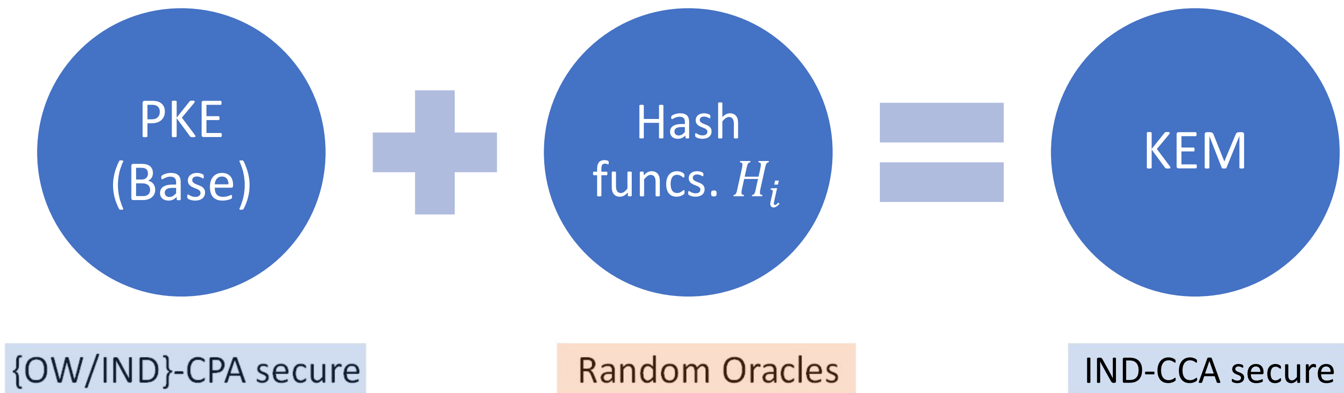


Fujisaki-Okamoto Transformation

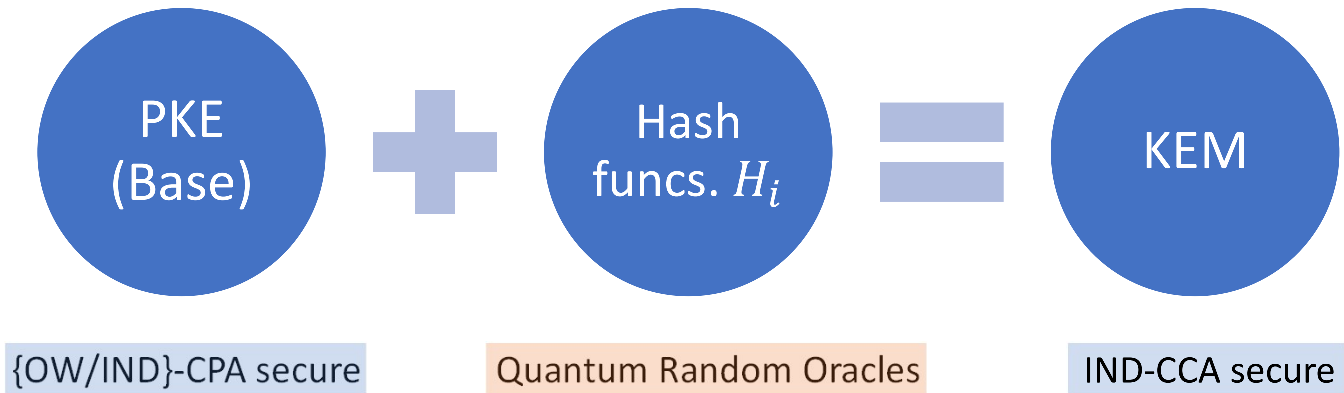
Fujisaki-Okamoto Transformation



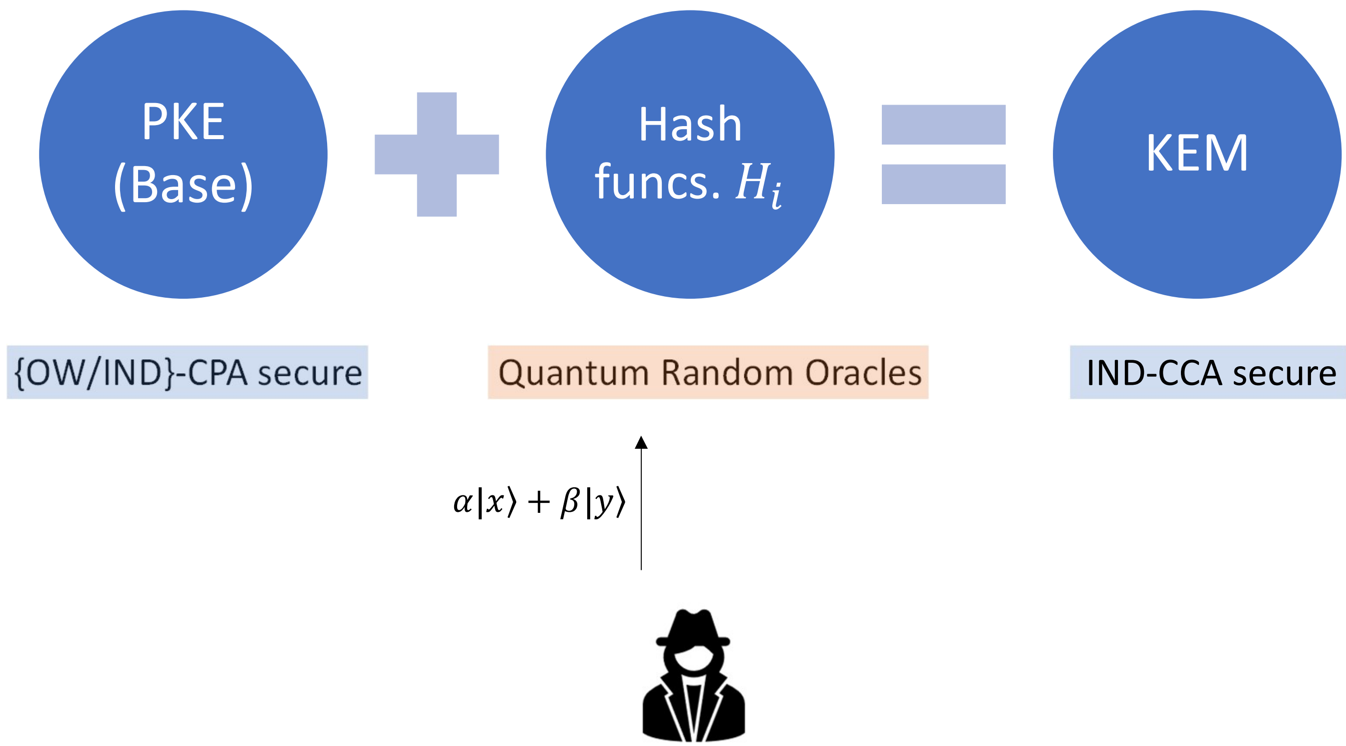
Fujisaki-Okamoto Transformation



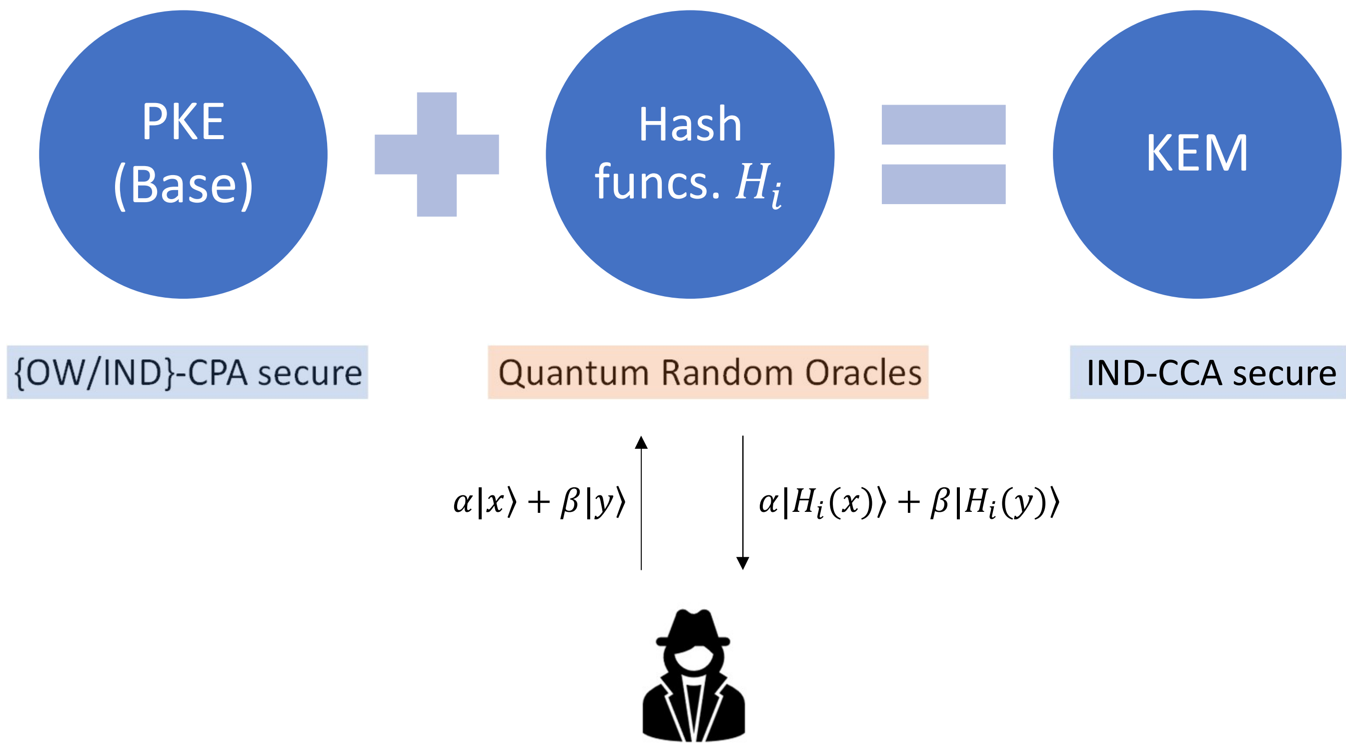
Fujisaki-Okamoto Transformation



Fujisaki-Okamoto Transformation



Fujisaki-Okamoto Transformation



Fujisaki-Okamoto Transformation

Classic McEliece

CRYSTALS-KYBER

SABER

NTRU

Fujisaki-Okamoto Transformation

Classic McEliece

CRYSTALS-KYBER

SABER

NTRU

KGen'	$\mathsf{Encap}(\mathsf{pk})$	$\mathsf{Decap}(\mathsf{sk}', c)$
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 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; G(m))$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : $\mathsf{sk}' = (\mathsf{sk}, s)$	3 : $k \leftarrow H(m, c)$	3 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; G(m'))$
4 : **return** $(\mathsf{pk}, \mathsf{sk}')$	4 : **return** (c, k)	4 : **if** $c' = c$ **then**
		5 : **return** $H(m', c)$
		6 : **else return** $H(s, c)$

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

Fujisaki-Okamoto Transformation

Classic McEliece

CRYSTALS-KYBER

SABER

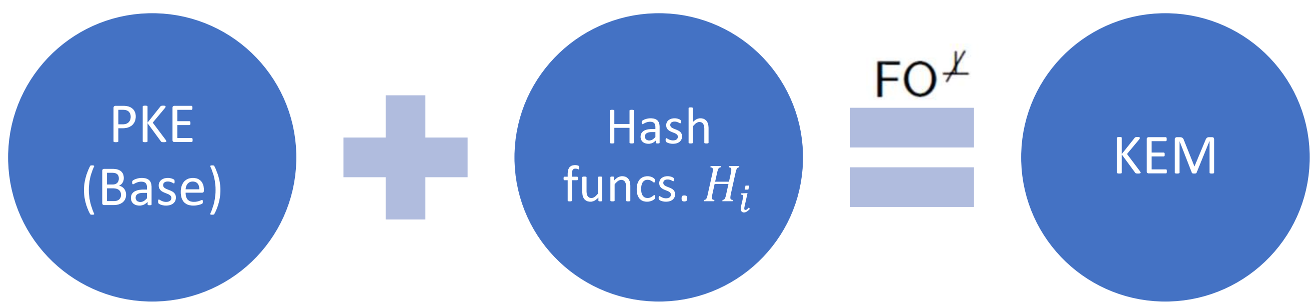
NTRU

FrodoKEM

KGen'	$\mathsf{Encap}(\mathsf{pk})$	$\mathsf{Decap}(\mathsf{sk}', c)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$	$m \leftarrow_{\$} \mathcal{M}$	Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
$s \leftarrow_{\$} \mathcal{M}$	$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; G(m))$	$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
$\mathsf{sk}' = (\mathsf{sk}, s)$	$k \leftarrow H(m, c)$	$c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; G(m'))$
return $(\mathsf{pk}, \mathsf{sk}')$	**return** (c, k)	**if** $c' = c$ **then**
		return $H(m', c)$
		else return $H(s, c)$

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

Anonymity from FO transforms



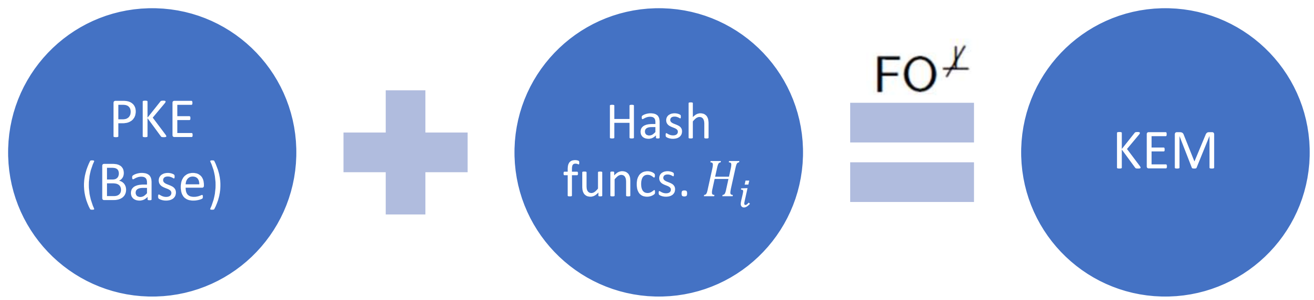
{OW/IND}-CPA secure

Quantum Random Oracles

IND-CCA secure

Shown in [Jiang-Zhang-Chen-Wang-Ma @Crypto’18]

Anonymity from FO transforms



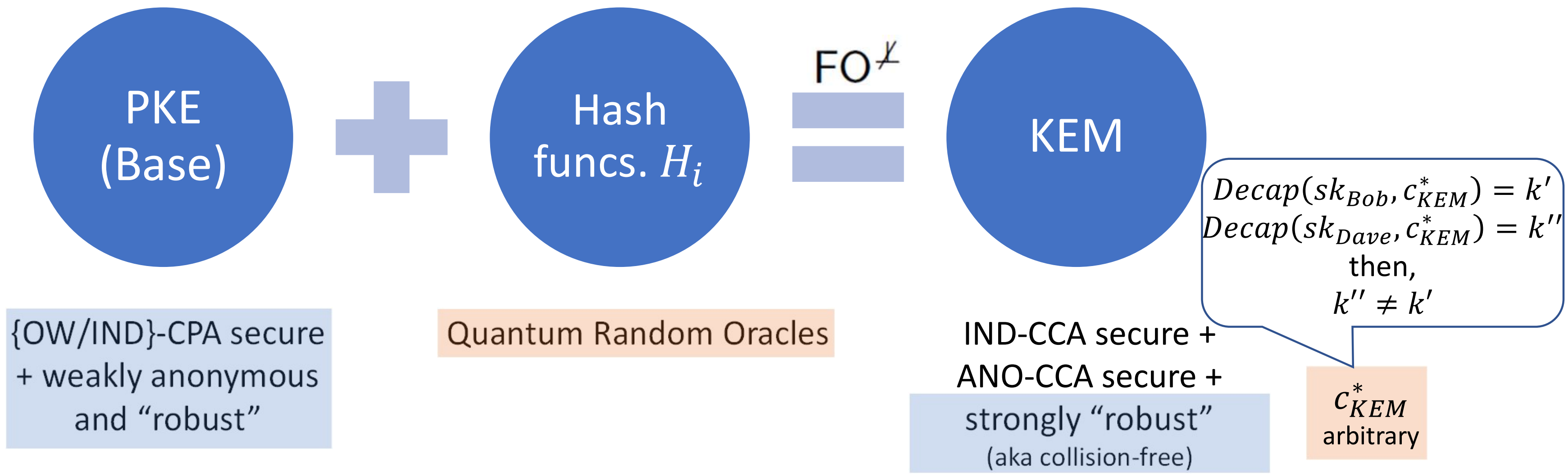
{OW/IND}-CPA secure
+ weakly anonymous
and "robust"

Quantum Random Oracles

IND-CCA secure +
ANO-CCA secure +
strongly "robust"
(aka collision-free)

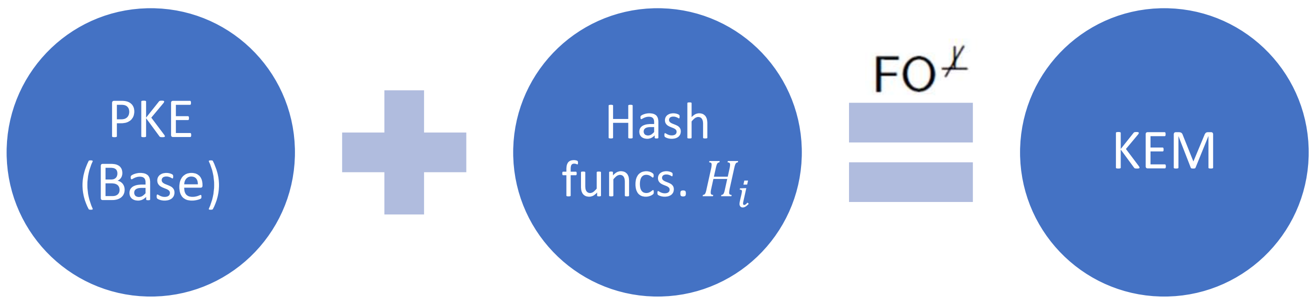
Shown in [Grubbs-Maram-Paterson
@Eurocrypt'22]

Anonymity from FO transforms



Shown in [Grubbs-Maram-Paterson @Eurocrypt'22]

Anonymity from FO transforms



{OW/IND}-CPA secure + weakly anonymous and "robust"

Quantum Random Oracles

IND-CCA secure + ANO-CCA secure + strongly "robust"
(aka collision-free)

Shown in [Grubbs-Maram-Paterson @Eurocrypt'22]

Extended Jiang et. al.'s proof techniques from a *single-key* setting (IND-CCA) to a *two-key* setting (ANO-CCA).

KEM-DEM Paradigm

Public-Key Encryption/KEMs

- Classic McEliece
- CRYSTALS-KYBER
- NTRU
- SABER

"Implicit-rejection" KEMs!

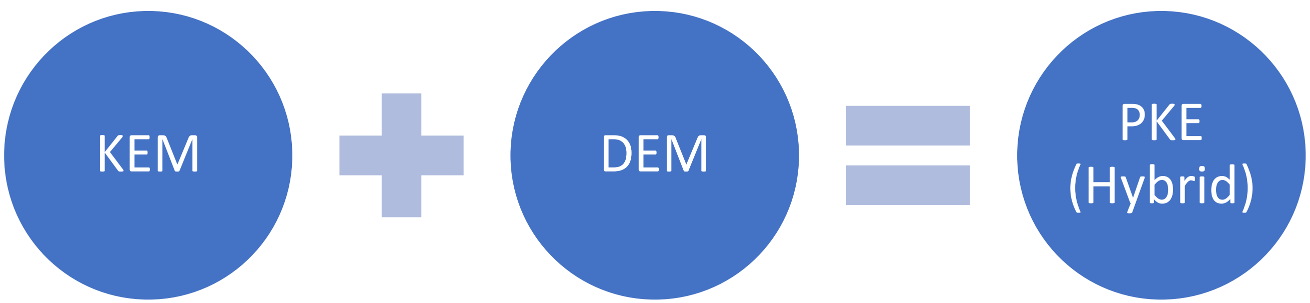
Cannot be even weakly robust.

Public-Key Encryption/KEMs

- BIKE
- FrodoKEM
- HQC
- NTRU Prime
- SIKE

Shown in [Grubbs-Maram-Paterson @Eurocrypt'22]; generalization of [Mohassel@Asiacrypt'10].

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + weakly robust

... is also necessary.

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure

KEM-DEM Paradigm

Public-Key Encryption/KEMs

- Classic McEliece
- CRYSTALS-KYBER
- NTRU
- SABER

"Implicit-rejection" KEMs!

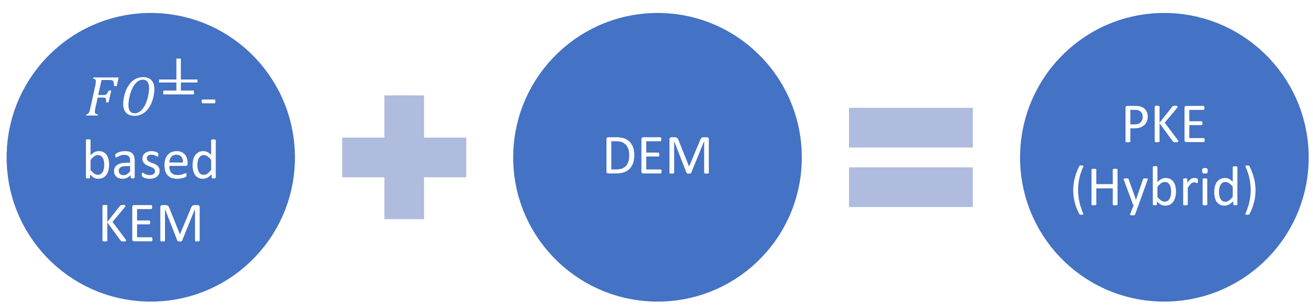
Cannot be even weakly robust.

Public-Key Encryption/KEMs

- BIKE
- FrodoKEM
- HQC
- NTRU Prime
- SIKE

Shown in [Grubbs-Maram-Paterson @Eurocrypt'22]; generalization of [Mohassel@Asiacrypt'10].

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + γ-spread base PKE

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure

KEM-DEM Paradigm

Public-Key Encryption/KEMs

Classic McEliece
CRYSTALS-KYBER
NTRU
SABER

"Implicit-rejection" KEMs!

Cannot be even weakly robust.

Public-Key Encryption/KEMs

BIKE
FrodoKEM
HQC
NTRU Prime
SIKE

Shown in [Grubbs-Maram-Paterson @Eurocrypt'22]; generalization of [Mohassel@Asiacrypt'10].

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$

$FO^{\pm}$-based KEM + DEM = PKE (Hybrid)

$c \leftarrow Enc^{base}(pk_{Bob}, m)$ should have large enough entropy.

$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + γ-spread base PKE

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

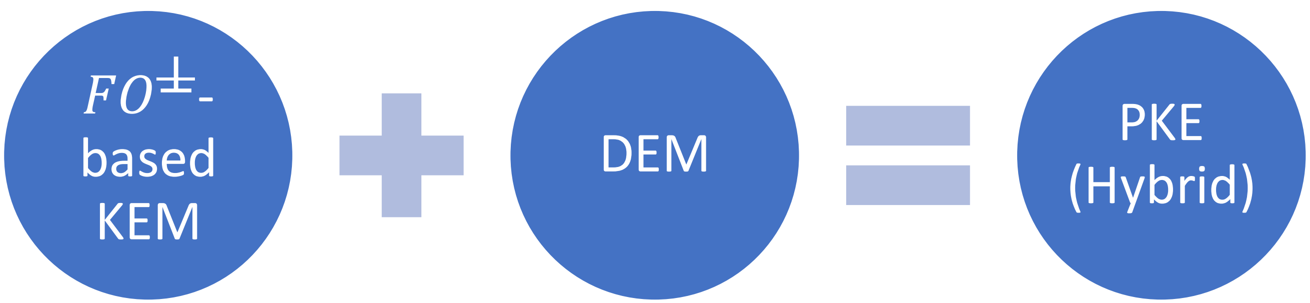
IND-CCA secure + ANO-CCA secure

Classic McEliece (CM)

Public-Key Encryption/KEMs	Public-Key Encryption/KEMs
Classic McEliece	BIKE
CRYSTALS-KYBER	FrodoKEM
NTRU	HQC
SABER	NTRU Prime
	SIKE

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$

$c \leftarrow Enc^{base}(pk_{Bob}, m)$ should have large enough entropy.



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + γ-spread base PKE

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

AE-secure

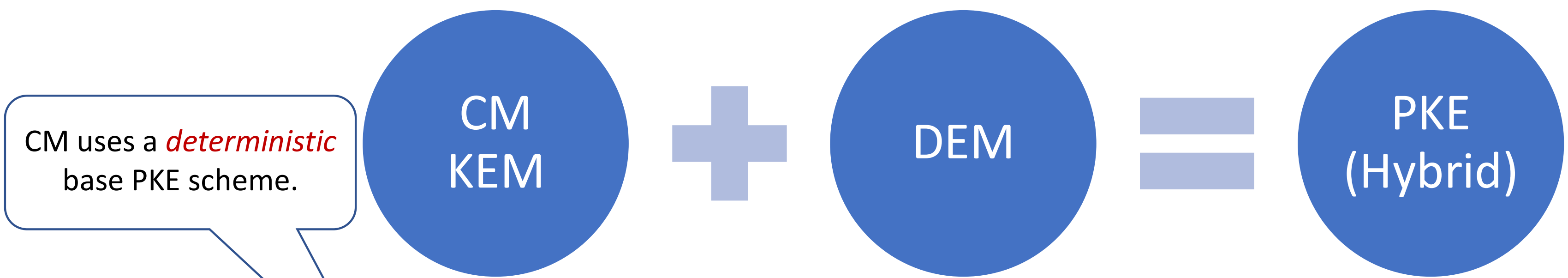
$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure

Classic McEliece (CM)

Public-Key Encryption/KEMs	Public-Key Encryption/KEMs
Classic McEliece	BIKE
CRYSTALS-KYBER	FrodoKEM
NTRU	HQC
SABER	NTRU Prime
	SIKE

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + γ-spread base PKE

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

AE-secure

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure

Classic McEliece (CM)

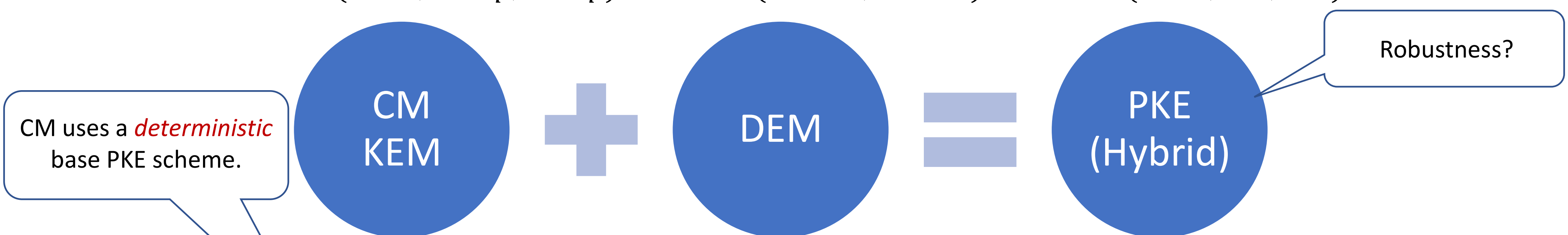
<u>Public-Key Encryption/KEMs</u>

- Classic McEliece
- CRYSTALS-KYBER
- NTRU
- SABER

<u>Public-Key Encryption/KEMs</u>

- BIKE
- FrodoKEM
- HQC
- NTRU Prime
- SIKE

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + γ-spread base PKE

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

AE-secure

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure

Classic McEliece (CM)

2.2.3 Encoding subroutine

The following algorithm ENCODE takes two inputs: a weight-t column vector $e \in \mathbb{F}_2^n$; and a public key T, i.e., an $(n-k) \times k$ matrix over $\mathbb{F}_2$. The algorithm output $\text{ENCODE}(e, T)$ is a vector $C_0 \in \mathbb{F}_2^{n-k}$. Here is the algorithm:

1. Define $H = (I_{n-k} \mid T)$.
2. Compute and return $C_0 = He \in \mathbb{F}_2^{n-k}$.

Classic McEliece (CM)

2.2.3 Encoding subroutine

The following algorithm ENCODE takes two inputs: a weight-t column vector $e \in \mathbb{F}_2^n$; and a public key T, i.e., an $(n-k) \times k$ matrix over $\mathbb{F}_2$. The algorithm output $\text{ENCODE}(e, T)$ is a vector $C_0 \in \mathbb{F}_2^{n-k}$. Here is the algorithm:

1. Define $H = (I_{n-k} \mid T)$.
2. Compute and return $C_0 = He \in \mathbb{F}_2^{n-k}$.

Classic McEliece (CM)

2.2.3 Encoding subroutine

The following algorithm ENCODE takes two inputs: a weight-t column vector $e \in \mathbb{F}_2^n$; and a public key T, i.e., an $(n-k) \times k$ matrix over $\mathbb{F}_2$. The algorithm output $\text{ENCODE}(e, T)$ is a vector $C_0 \in \mathbb{F}_2^{n-k}$. Here is the algorithm:

1. Define $H = (I_{n-k} \mid T)$.
2. Compute and return $C_0 = He \in \mathbb{F}_2^{n-k}$.

Classic McEliece (CM)

2.2.3 Encoding subroutine

The following algorithm ENCODE takes two inputs: a weight-t column vector $e \in \mathbb{F}_2^n$; and a public key T, i.e., an $(n-k) \times k$ matrix over $\mathbb{F}_2$. The algorithm output ENCODE(e, T) is a vector $C_0 \in \mathbb{F}_2^{n-k}$. Here is the algorithm:

1. Define $H = (I_{n-k} \mid T)$.
2. Compute and return $C_0 = He \in \mathbb{F}_2^{n-k}$.

Classic McEliece (CM)

2.2.3 Encoding subroutine

The following algorithm Encode takes two inputs: a weight-t column vector $e \in \mathbb{F}_2^n$; and a public key T, i.e., an $(n-k) \times k$ matrix over $\mathbb{F}_2$. The algorithm output $\text{Encode}(e, T)$ is a vector $C_0 \in \mathbb{F}_2^{n-k}$. Here is the algorithm:

1. Define $H = (I_{n-k} \mid T)$.
2. Compute and return $C_0 = He \in \mathbb{F}_2^{n-k}$.

Classic McEliece (CM)

2.2.3 Encoding subroutine

The following algorithm ENCODE takes two inputs: a weight-t column vector $e \in \mathbb{F}_2^n$; and a public key T, i.e., an $(n-k) \times k$ matrix over $\mathbb{F}_2$. The algorithm output ENCODE(e, T) is a vector $C_0 \in \mathbb{F}_2^{n-k}$. Here is the algorithm:

1. Define $H = (I_{n-k} \mid T)$.
2. Compute and return $C_0 = He \in \mathbb{F}_2^{n-k}$.

Fix any "message" $e = \binom{e_{n-k}}{0^k}$:

Classic McEliece (CM)

2.2.3 Encoding subroutine

The following algorithm ENCODE takes two inputs: a weight-t column vector $e \in \mathbb{F}_2^n$; and a public key T, i.e., an $(n-k) \times k$ matrix over $\mathbb{F}_2$. The algorithm output ENCODE(e, T) is a vector $C_0 \in \mathbb{F}_2^{n-k}$. Here is the algorithm:

1. Define $H = (I_{n-k} \mid T)$.
2. Compute and return $C_0 = He \in \mathbb{F}_2^{n-k}$.

Fix any "message" $e = \binom{e_{n-k}}{0^k}$:

- $(n-k \ \geq t$ in all CM parameters)

Classic McEliece (CM)

2.2.3 Encoding subroutine

The following algorithm ENCODE takes two inputs: a weight-t column vector $e \in \mathbb{F}_2^n$; and a public key T, i.e., an $(n-k) \times k$ matrix over $\mathbb{F}_2$. The algorithm output $\text{ENCODE}(e, T)$ is a vector $C_0 \in \mathbb{F}_2^{n-k}$. Here is the algorithm:

1. Define $H = (I_{n-k} \mid T)$.
2. Compute and return $C_0 = He \in \mathbb{F}_2^{n-k}$.

Fix any "message" $e = \binom{e_{n-k}}{0^k}$:

- $(n-k \geq t$ in all CM parameters)
- $C_0 = (I_{n-k}|T)\binom{e_{n-k}}{0^k} = e_{n-k}$ – i.e., independent of public-key T.

Classic McEliece (CM)

2.2.3 Encoding subroutine

The following algorithm ENCODE takes two inputs: a weight-t column vector $e \in \mathbb{F}_2^n$; and a public key T, i.e., an $(n-k) \times k$ matrix over $\mathbb{F}_2$. The algorithm output ENCODE(e, T) is a vector $C_0 \in \mathbb{F}_2^{n-k}$. Here is the algorithm:

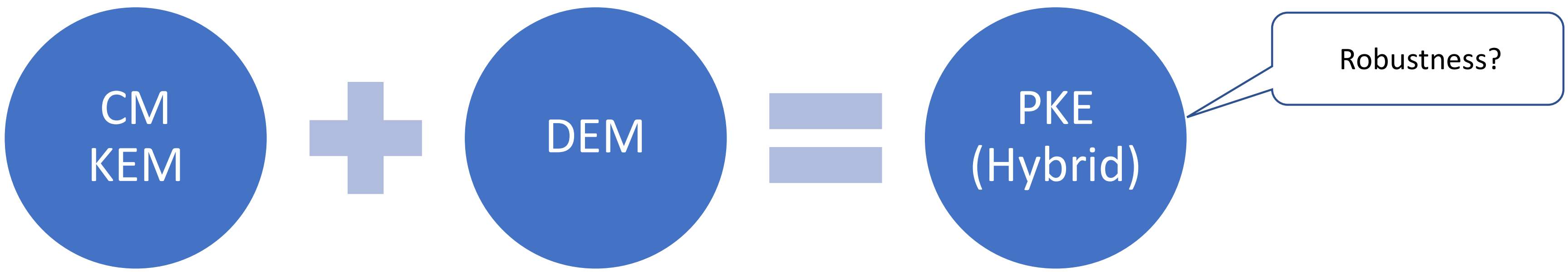
1. Define $H = (I_{n-k} \mid T)$.
2. Compute and return $C_0 = He \in \mathbb{F}_2^{n-k}$.

Fix any "message" $e = \binom{e_{n-k}}{0^k}$:

- ($n - k \geq t$ in all CM parameters)
- $C_0 = (I_{n-k}|T)\binom{e_{n-k}}{0^k} = e_{n-k}$ – i.e., independent of public-key T.
- Because of perfect correctness, C_0 must decrypt to fixed e under *any private key* of CM's base PKE scheme.

Classic McEliece (CM)

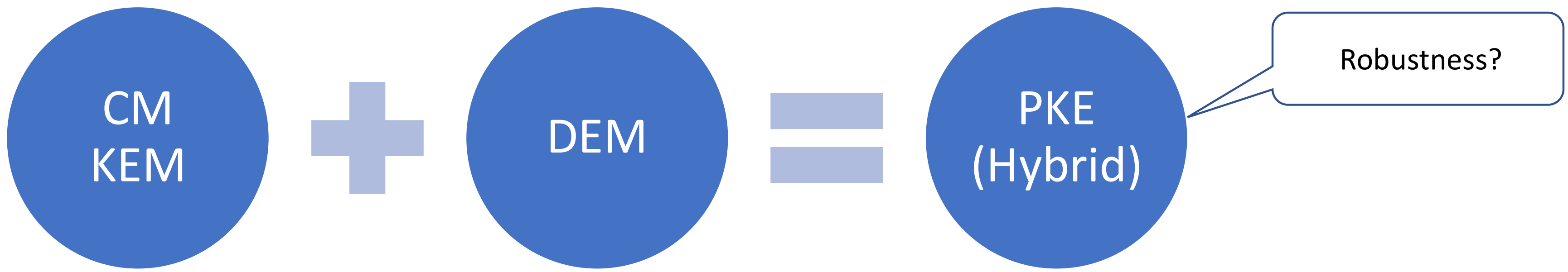
$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$ $c_{DEM} \leftarrow Enc^{sym}(k, m)$ $(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

Classic McEliece (CM)

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$ $c_{DEM} \leftarrow Enc^{sym}(k, m)$ $(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

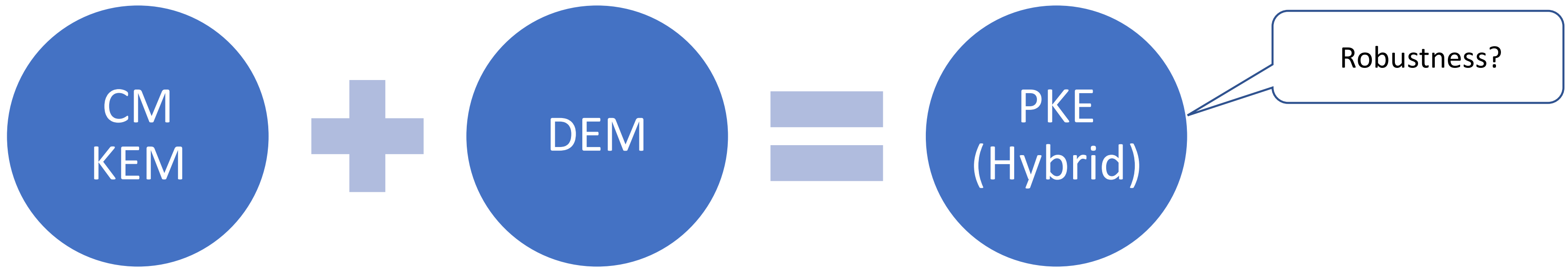
2.4.5 Encapsulation

The following randomized algorithm ENCAP takes as input a public key T. It outputs a ciphertext C and a session key K. Here is the algorithm:

1. Use FIXEDWEIGHT to generate a vector $e \in \mathbb{F}_2^n$ of weight t.
2. Compute $C_0 = \text{ENCODE}(e, T)$.
3. Compute $C_1 = \mathsf{H}(2, e)$; see Section 2.5.2 for H input encodings. Put $C = (C_0, C_1)$.
4. Compute $K = \mathsf{H}(1, e, C)$; see Section 2.5.2 for H input encodings.
5. Output ciphertext C and session key K.

Classic McEliece (CM)

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$ $c_{DEM} \leftarrow Enc^{sym}(k, m)$ $(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

2.4.5 Encapsulation

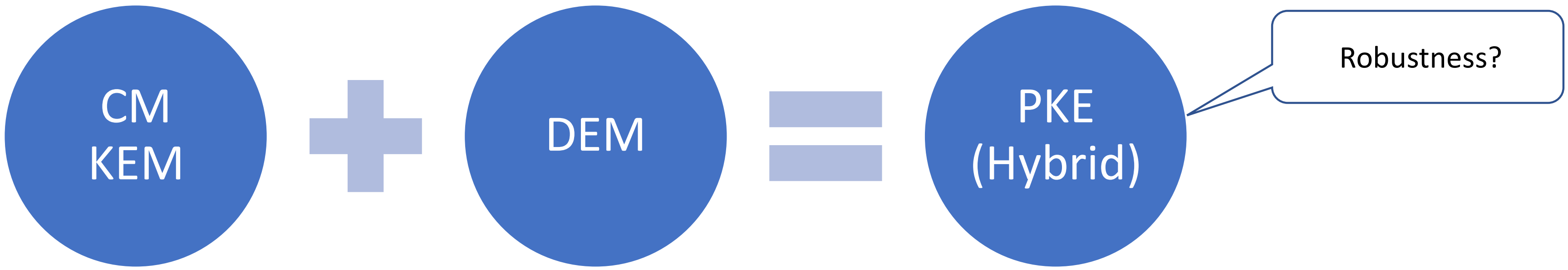
The following randomized algorithm ENCAP takes as input a public key T. It outputs a ciphertext C and a session key K. Here is the algorithm:

1. Use FIXEDWEIGHT to generate a vector $e \in \mathbb{F}_2^n$ of weight t.
2. Compute $C_0 = \text{ENCODE}(e, T)$.
3. Compute $C_1 = \mathsf{H}(2, e)$; see Section 2.5.2 for H input encodings. Put $C = (C_0, C_1)$.
4. Compute $K = \mathsf{H}(1, e, C)$; see Section 2.5.2 for H input encodings.
5. Output ciphertext C and session key K.

For *any* message m:

Classic McEliece (CM)

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$ $c_{DEM} \leftarrow Enc^{sym}(k, m)$ $(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

2.4.5 Encapsulation

The following randomized algorithm ENCAP takes as input a public key T. It outputs a ciphertext C and a session key K. Here is the algorithm:

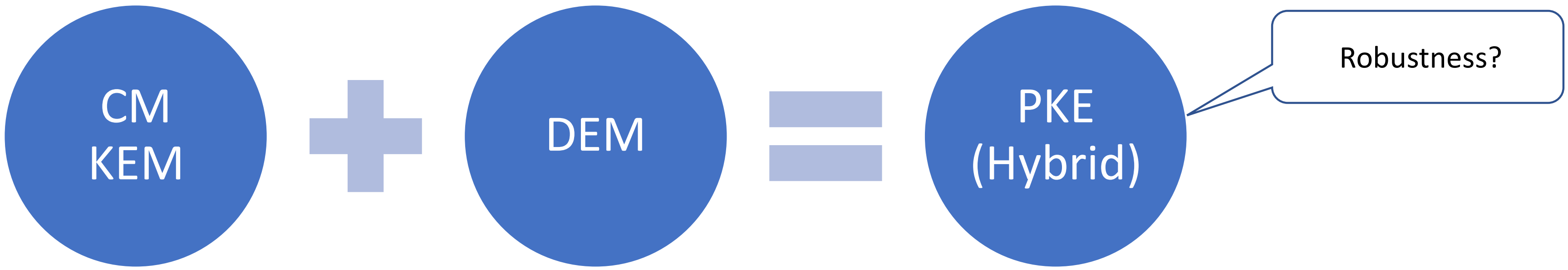
1. Use FIXEDWEIGHT to generate a vector $e \in \mathbb{F}_2^n$ of weight t.
2. Compute $C_0 = \text{ENCODE}(e, T)$.
3. Compute $C_1 = \mathsf{H}(2, e)$; see Section 2.5.2 for H input encodings. Put $C = (C_0, C_1)$.
4. Compute $K = \mathsf{H}(1, e, C)$; see Section 2.5.2 for H input encodings.
5. Output ciphertext C and session key K.

For *any* message m:

- Fix vector $e = \binom{e_{n-k}}{0^k}$.

Classic McEliece (CM)

$KEM = (KGen, Encap, Decap)$ $\quad DEM = (Enc^{sym}, Dec^{sym})$ $\quad PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$ $\quad c_{DEM} \leftarrow Enc^{sym}(k, m)$ $\quad (c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

2.4.5 Encapsulation

The following randomized algorithm ENCAP takes as input a public key T. It outputs a ciphertext C and a session key K. Here is the algorithm:

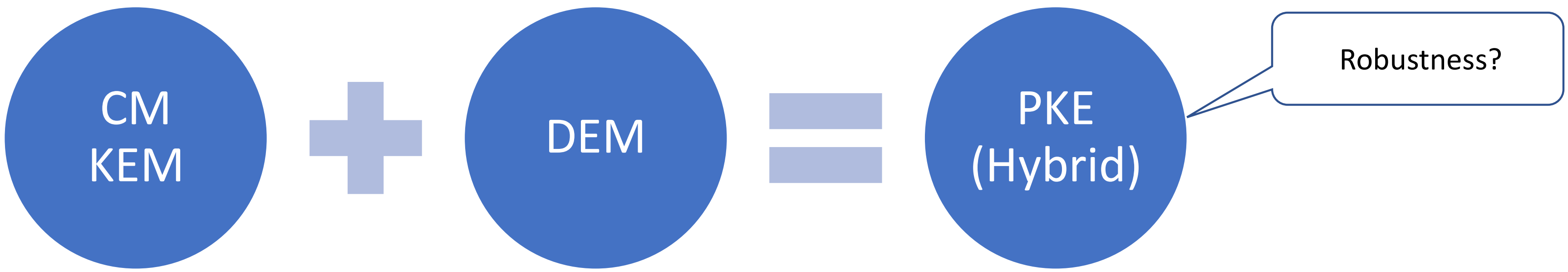
1. Use FIXEDWEIGHT to generate a vector $e \in \mathbb{F}_2^n$ of weight t.
2. Compute $C_0 = \text{ENCODE}(e, T)$.
3. Compute $C_1 = \mathsf{H}(2, e)$; see Section 2.5.2 for H input encodings. Put $C = (C_0, C_1)$.
4. Compute $K = \mathsf{H}(1, e, C)$; see Section 2.5.2 for H input encodings.
5. Output ciphertext C and session key K.

For *any* message m:

- Fix vector $e = \binom{e_{n-k}}{0^k}$.
- Set $C_0 = e_{n-k}, C_1 = \mathsf{H}(2, e)$ and $c_{KEM} \leftarrow (C_0, C_1)$.

Classic McEliece (CM)

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$ $c_{DEM} \leftarrow Enc^{sym}(k, m)$ $(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

2.4.5 Encapsulation

The following randomized algorithm ENCAP takes as input a public key T. It outputs a ciphertext C and a session key K. Here is the algorithm:

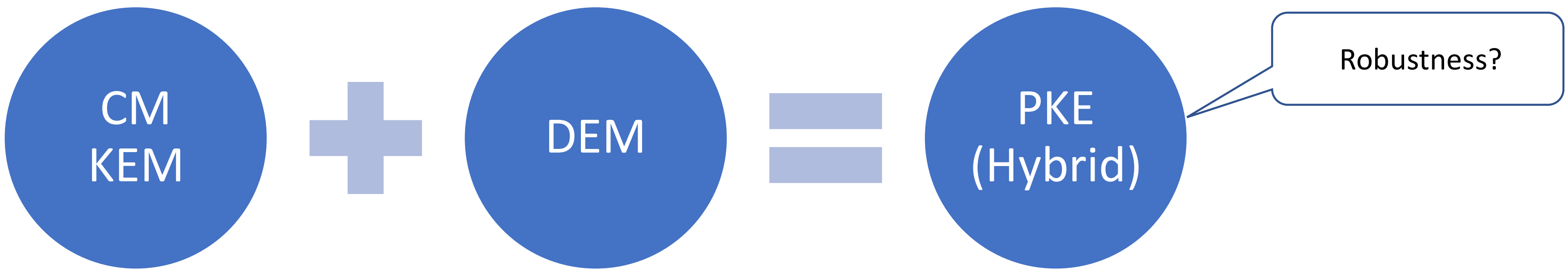
1. Use FIXEDWEIGHT to generate a vector $e \in \mathbb{F}_2^n$ of weight t.
2. Compute $C_0 = \text{ENCODE}(e, T)$.
3. Compute $C_1 = \mathsf{H}(2, e)$; see Section 2.5.2 for H input encodings. Put $C = (C_0, C_1)$.
4. Compute $K = \mathsf{H}(1, e, C)$; see Section 2.5.2 for H input encodings.
5. Output ciphertext C and session key K.

For *any* message m:

- Fix vector $e = \binom{e_{n-k}}{0^k}$.
- Set $C_0 = e_{n-k}, C_1 = \mathsf{H}(2, e)$ and $c_{KEM} \leftarrow (C_0, C_1)$.
- Compute $k = \mathsf{H}(1, e, c_{KEM})$ and $c_{DE} \leftarrow Enc^{sym}(k, m)$.

Classic McEliece (CM)

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$ $c_{DEM} \leftarrow Enc^{sym}(k, m)$ $(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

2.4.5 Encapsulation

The following randomized algorithm ENCAP takes as input a public key T. It outputs a ciphertext C and a session key K. Here is the algorithm:

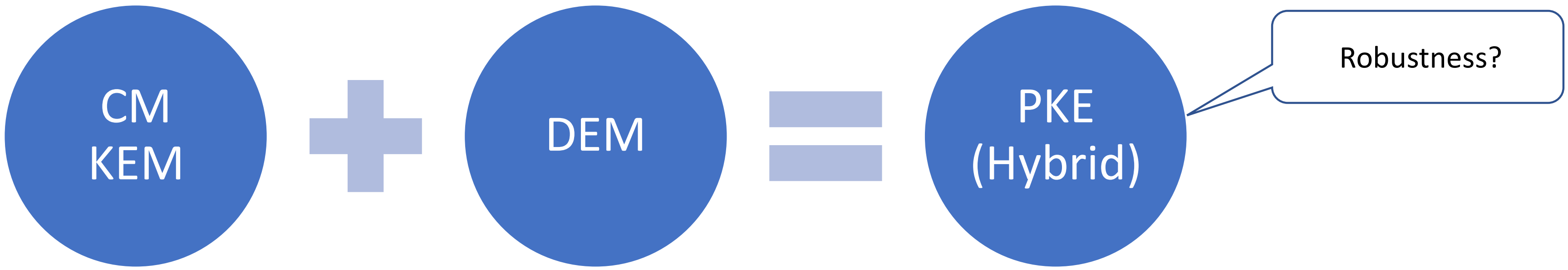
1. Use FIXEDWEIGHT to generate a vector $e \in \mathbb{F}_2^n$ of weight t.
2. Compute $C_0 = \text{ENCODE}(e, T)$.
3. Compute $C_1 = \mathsf{H}(2, e)$; see Section 2.5.2 for H input encodings. Put $C = (C_0, C_1)$.
4. Compute $K = \mathsf{H}(1, e, C)$; see Section 2.5.2 for H input encodings.
5. Output ciphertext C and session key K.

For *any* message m:

- Fix vector $e = \binom{e_{n-k}}{0^k}$.
- Set $C_0 = e_{n-k}, C_1 = \mathsf{H}(2, e)$ and $c_{KEM} \leftarrow (C_0, C_1)$.
- Compute $k = \mathsf{H}(1, e, c_{KEM})$ and $c_{DEM} \leftarrow Enc^{sym}(k, m)$.
- Return $c \leftarrow (c_{KEM}, c_{DEM})$.

Classic McEliece (CM)

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$ $c_{DEM} \leftarrow Enc^{sym}(k, m)$ $(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

2.4.5 Encapsulation

The following randomized algorithm ENCAP takes as input a public key T. It outputs a ciphertext C and a session key K. Here is the algorithm:

1. Use FIXEDWEIGHT to generate a vector $e \in \mathbb{F}_2^n$ of weight t.
2. Compute $C_0 = \text{ENCODE}(e, T)$.
3. Compute $C_1 = \mathsf{H}(2, e)$; see Section 2.5.2 for H input encodings. Put $C = (C_0, C_1)$.
4. Compute $K = \mathsf{H}(1, e, C)$; see Section 2.5.2 for H input encodings.
5. Output ciphertext C and session key K.

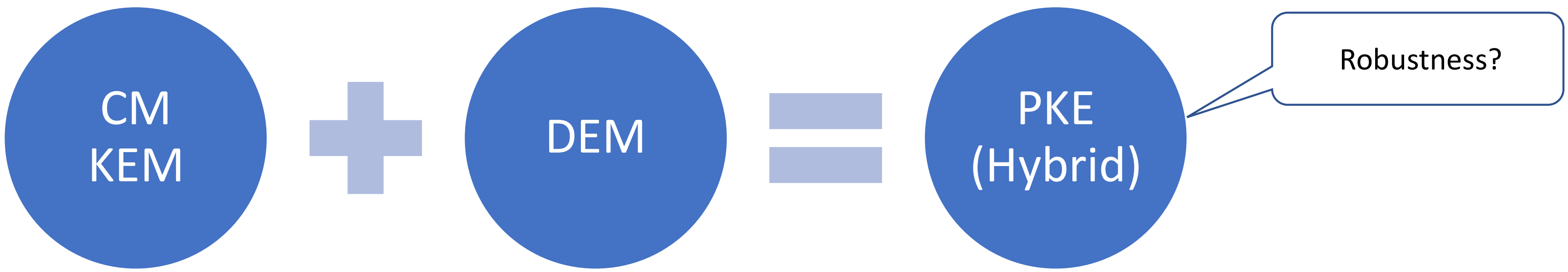
For *any* message m:

- Fix vector $e = \binom{e_{n-k}}{0^k}$.
- Set $C_0 = e_{n-k}, C_1 = \mathsf{H}(2, e)$ and $c_{KEM} \leftarrow (C_0, C_1)$.
- Compute $k = \mathsf{H}(1, e, c_{KEM})$ and $c_{DEM} \leftarrow Enc^{sym}(k, m)$.
- Return $c \leftarrow (c_{KEM}, c_{DEM})$.

For *any* CM private key sk_* ,

Classic McEliece (CM)

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$ $c_{DEM} \leftarrow Enc^{sym}(k, m)$ $(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

2.4.5 Encapsulation

The following randomized algorithm ENCAP takes as input a public key T. It outputs a ciphertext C and a session key K. Here is the algorithm:

1. Use FIXEDWEIGHT to generate a vector $e \in \mathbb{F}_2^n$ of weight t.
2. Compute $C_0 = \text{ENCODE}(e, T)$.
3. Compute $C_1 = \mathsf{H}(2, e)$; see Section 2.5.2 for H input encodings. Put $C = (C_0, C_1)$.
4. Compute $K = \mathsf{H}(1, e, C)$; see Section 2.5.2 for H input encodings.
5. Output ciphertext C and session key K.

For *any* message m:

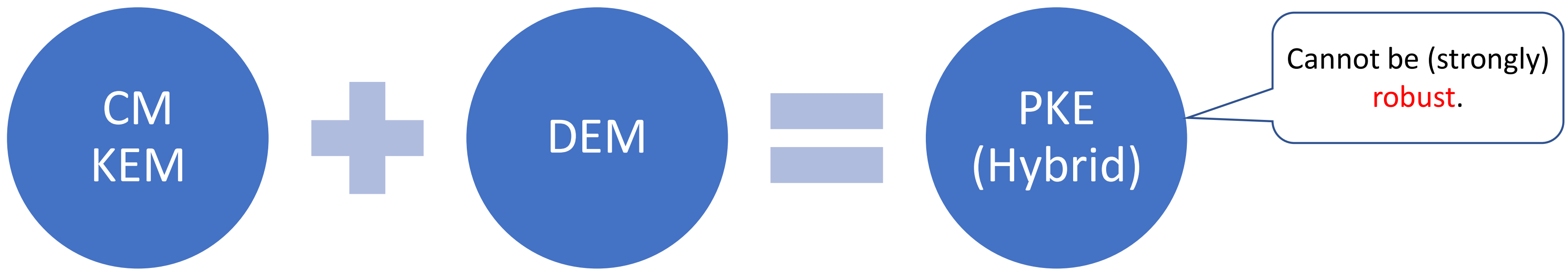
- Fix vector $e = \binom{e_{n-k}}{0^k}$.
- Set $C_0 = e_{n-k}, C_1 = \mathsf{H}(2, e)$ and $c_{KEM} \leftarrow (C_0, C_1)$.
- Compute $k = \mathsf{H}(1, e, c_{KEM})$ and $c_{DE} \leftarrow Enc^{sym}(k, m)$.
- Return $c \leftarrow (c_{KEM}, c_{DEM})$.

For *any* CM private key sk_*,

$Dec(sk_*, c) = m \ (\neq \perp)$.

Classic McEliece (CM)

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$ $c_{DEM} \leftarrow Enc^{sym}(k, m)$ $(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

2.4.5 Encapsulation

The following randomized algorithm ENCAP takes as input a public key T. It outputs a ciphertext C and a session key K. Here is the algorithm:

1. Use FIXEDWEIGHT to generate a vector $e \in \mathbb{F}_2^n$ of weight t.
2. Compute $C_0 = \text{ENCODE}(e, T)$.
3. Compute $C_1 = \mathsf{H}(2, e)$; see Section 2.5.2 for H input encodings. Put $C = (C_0, C_1)$.
4. Compute $K = \mathsf{H}(1, e, C)$; see Section 2.5.2 for H input encodings.
5. Output ciphertext C and session key K.

For *any* message m:

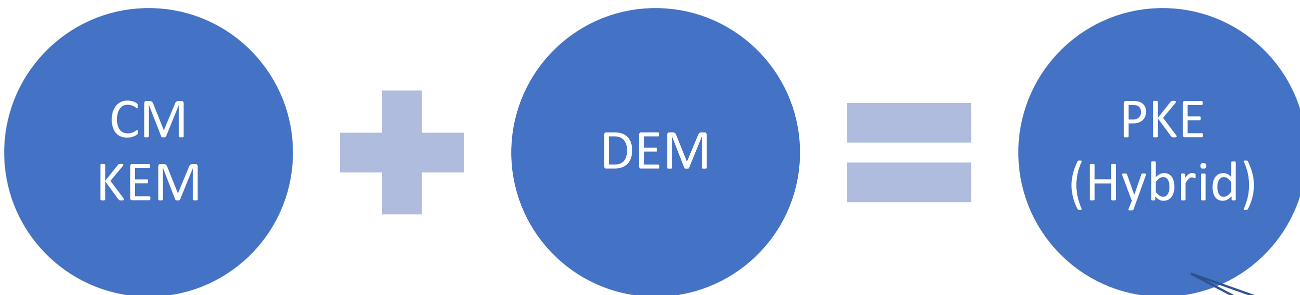
- Fix vector $e = \binom{e_{n-k}}{0^k}$.
- Set $C_0 = e_{n-k}, C_1 = \mathsf{H}(2, e)$ and $c_{KEM} \leftarrow (C_0, C_1)$.
- Compute $k = \mathsf{H}(1, e, c_{KEM})$ and $c_{DE} \leftarrow Enc^{sym}(k, m)$.
- Return $c \leftarrow (c_{KEM}, c_{DEM})$.

For *any* CM private key sk_*,

$$Dec(sk_*, c) = m \ (\neq \perp).$$

Classic McEliece (CM)

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



Cannot be (strongly) robust.

$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$ $c_{DEM} \leftarrow Enc^{sym}(k, m)$ $(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

But can be ANO-CCA secure.
[Xagawa@Eurocrypt'22]

2.4.5 Encapsulation

The following randomized algorithm ENCAP takes as input a public key T. It outputs a ciphertext C and a session key K. Here is the algorithm:

1. Use FIXEDWEIGHT to generate a vector $e \in \mathbb{F}_2^n$ of weight t.
2. Compute $C_0 = \text{ENCODE}(e, T)$.
3. Compute $C_1 = \mathsf{H}(2, e)$; see Section 2.5.2 for H input encodings. Put $C = (C_0, C_1)$.
4. Compute $K = \mathsf{H}(1, e, C)$; see Section 2.5.2 for H input encodings.
5. Output ciphertext C and session key K.

For *any* message m:

- Fix vector $e = \binom{e_{n-k}}{0^k}$.
- Set $C_0 = e_{n-k}, C_1 = \mathsf{H}(2, e)$ and $c_{KEM} \leftarrow (C_0, C_1)$.
- Compute $k = \mathsf{H}(1, e, c_{KEM})$ and $c_{DEM} \leftarrow Enc^{sym}(k, m)$.
- Return $c \leftarrow (c_{KEM}, c_{DEM})$.

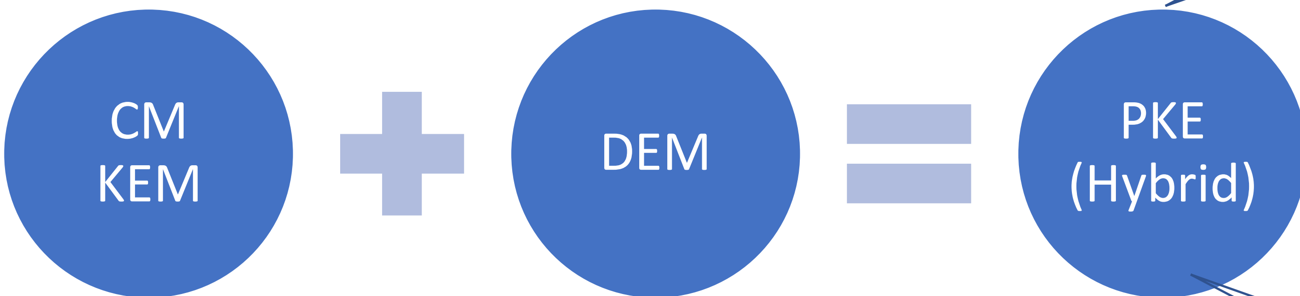
For *any* CM private key sk_*,

$$Dec(sk_*, c) = m \; (\neq \perp).$$

Classic McEliece (CM)

Xagawa relied on a stronger *single-key* notion, i.e., strong pseudo-randomness.

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



Cannot be (strongly) robust.

$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$ $c_{DEM} \leftarrow Enc^{sym}(k, m)$ $(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

But can be ANO-CCA secure. [Xagawa@Eurocrypt'22]

2.4.5 Encapsulation

The following randomized algorithm ENCAP takes as input a public key T. It outputs a ciphertext C and a session key K. Here is the algorithm:

1. Use FIXEDWEIGHT to generate a vector $e \in \mathbb{F}_2^n$ of weight t.
2. Compute $C_0 = \text{ENCODE}(e, T)$.
3. Compute $C_1 = \mathsf{H}(2, e)$; see Section 2.5.2 for H input encodings. Put $C = (C_0, C_1)$.
4. Compute $K = \mathsf{H}(1, e, C)$; see Section 2.5.2 for H input encodings.
5. Output ciphertext C and session key K.

For *any* message m:

- Fix vector $e = \binom{e_{n-k}}{0^k}$.
- Set $C_0 = e_{n-k}, C_1 = \mathsf{H}(2, e)$ and $c_{KEM} \leftarrow (C_0, C_1)$.
- Compute $k = \mathsf{H}(1, e, c_{KEM})$ and $c_{DEM} \leftarrow Enc^{sym}(k, m)$.
- Return $c \leftarrow (c_{KEM}, c_{DEM})$.

For *any* CM private key sk_*,

$Dec(sk_*, c) = m\ (\neq \perp)$.

CRYSTALS-KYBER and SABER

Public-Key Encryption/KEMs	Public-Key Encryption/KEMs
Classic McEliece	BIKE
CRYSTALS-KYBER	FrodoKEM
NTRU	HQC
SABER	NTRU Prime
	SIKE

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$

$FO^{\pm}$-based KEM + DEM = PKE (Hybrid)

$c \leftarrow Enc^{base}(pk_{Bob}, m)$ should have large enough entropy.

$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + γ-spread base PKE

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

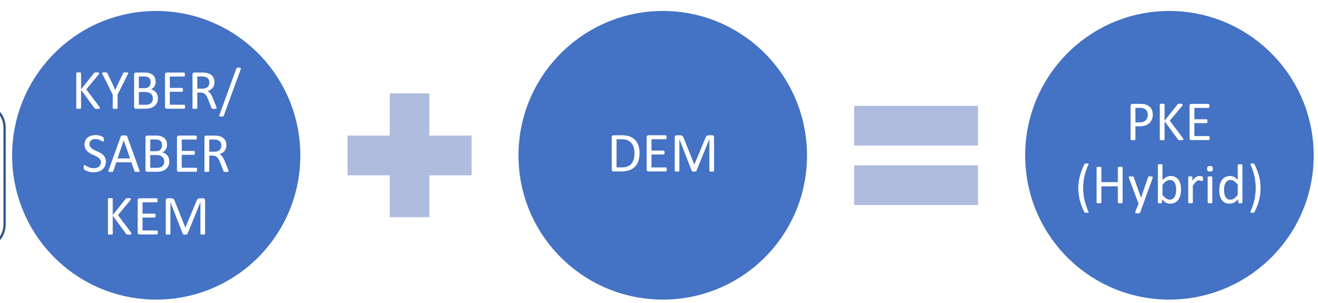
IND-CCA secure + ANO-CCA secure

CRYSTALS-KYBER and SABER

Public-Key Encryption/KEMs	Public-Key Encryption/KEMs
Classic McEliece	BIKE
CRYSTALS-KYBER	FrodoKEM
NTRU	HQC
SABER	NTRU Prime
	SIKE

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$

$c \leftarrow Enc^{base}(pk_{Bob}, m)$ should have large enough entropy.



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + γ-spread base PKE

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure

CRYSTALS-KYBER and SABER

<u>Public-Key Encryption/KEMs</u>

Classic McEliece

CRYSTALS-KYBER

NTRU

SABER

<u>Public-Key Encryption/KEMs</u>

BIKE

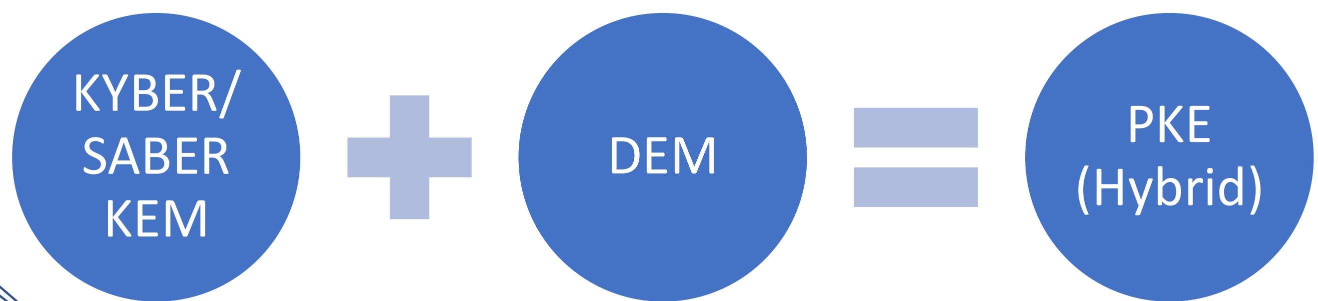
FrodoKEM

HQC

NTRU Prime

SIKE

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



?

$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + γ-spread base PKE

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure

CRYSTALS-KYBER and SABER

<u>Public-Key Encryption/KEMs</u>

Classic McEliece

CRYSTALS-KYBER

NTRU

SABER

<u>Public-Key Encryption/KEMs</u>

BIKE

FrodoKEM

HQC

NTRU Prime

SIKE

KGen'	$\mathsf{Encap}(\mathsf{pk})$	$\mathsf{Decap}(\mathsf{sk}', c)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$	$m \leftarrow_\$ \mathcal{M}$	Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
$s \leftarrow_\$ \mathcal{M}$	$r \leftarrow G(m)$	$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
$\mathsf{sk}' = (\mathsf{sk}, s)$	$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	$r' \leftarrow G(m')$
return $(\mathsf{pk}, \mathsf{sk}')$	$k \leftarrow H(m, c)$	$c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	return (c, k)	**if** $c' = c$ **then**
		return $H(m', c)$
		else return $H(s, c)$

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

CRYSTALS-KYBER and SABER

<u>Public-Key Encryption/KEMs</u>

- Classic McEliece
- CRYSTALS-KYBER
- NTRU
- SABER

<u>Public-Key Encryption/KEMs</u>

- BIKE
- FrodoKEM
- HQC
- NTRU Prime
- SIKE

KGen'	$\mathsf{Encap}(\mathsf{pk})$	$\mathsf{Decap}(\mathsf{sk}', c)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$	$m \leftarrow_{\$} \mathcal{M}$	Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
$s \leftarrow_{\$} \mathcal{M}$	$r \leftarrow G(m)$	$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
$\mathsf{sk}' = (\mathsf{sk}, s)$	$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	$r' \leftarrow G(m')$
return $(\mathsf{pk}, \mathsf{sk}')$	$k \leftarrow H(m, c)$	$c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	return (c, k)	**if** $c' = c$ **then**
		return $H(m', c)$
		else return $H(s, c)$

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

KGen'	$\mathsf{Encap}(\mathsf{pk})$	$\mathsf{Decap}(\mathsf{sk}', c)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$	$m \leftarrow_{\$} \mathcal{M}$	Parse $\mathsf{sk}' = (\mathsf{sk}, \mathsf{pk}, F(\mathsf{pk}), s)$
$s \leftarrow_{\$} \mathcal{M}$	$m \leftarrow F(m)$	$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
$\mathsf{sk}' \leftarrow (\mathsf{sk}, \mathsf{pk}, F(\mathsf{pk}), s)$	$(\hat{k}, r) \leftarrow G(F(pk), m)$	$(\hat{k}', r') \leftarrow G(F(\mathsf{pk}), m')$
return $(\mathsf{pk}, \mathsf{sk}')$	$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	$c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	$k \leftarrow \mathsf{KDF}(\hat{k}, F(c))$	**if** $c' = c$ **then**
	return (c, k)	**return** $\mathsf{KDF}(\hat{k}', F(c))$
		else return $\mathsf{KDF}(s, F(c))$

CRYSTALS-KYBER, Saber

CRYSTALS-KYBER and SABER

<u>Public-Key Encryption/KEMs</u>

Classic McEliece

CRYSTALS-KYBER

NTRU

SABER

<u>Public-Key Encryption/KEMs</u>

BIKE

FrodoKEM

HQC

NTRU Prime

SIKE

"$k \leftarrow H(m, c)$"

KGen'	$\mathsf{Encap}(\mathsf{pk})$	$\mathsf{Decap}(\mathsf{sk}', c)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$	$m \leftarrow_{\$} \mathcal{M}$	Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
$s \leftarrow_{\$} \mathcal{M}$	$r \leftarrow G(m)$	$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
$\mathsf{sk}' = (\mathsf{sk}, s)$	$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	$r' \leftarrow G(m')$
return $(\mathsf{pk}, \mathsf{sk}')$	$k \leftarrow H(m, c)$	$c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	return (c, k)	**if** $c' = c$ **then**
		return $H(m', c)$
		else return $H(s, c)$

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

"$k \leftarrow H(G(m), F(c))$"

KGen'	$\mathsf{Encap}(\mathsf{pk})$	$\mathsf{Decap}(\mathsf{sk}', c)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$	$m \leftarrow_{\$} \mathcal{M}$	Parse $\mathsf{sk}' = (\mathsf{sk}, \mathsf{pk}, F(\mathsf{pk}), s)$
$s \leftarrow_{\$} \mathcal{M}$	$m \leftarrow F(m)$	$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
$\mathsf{sk}' \leftarrow (\mathsf{sk}, \mathsf{pk}, F(\mathsf{pk}), s)$	$(\hat{k}, r) \leftarrow G(F(pk), m)$	$(\hat{k}', r') \leftarrow G(F(\mathsf{pk}), m')$
return $(\mathsf{pk}, \mathsf{sk}')$	$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	$c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	$k \leftarrow \mathsf{KDF}(\hat{k}, F(c))$	**if** $c' = c$ **then**
	return (c, k)	**return** $\mathsf{KDF}(\hat{k}', F(c))$
		else return $\mathsf{KDF}(s, F(c))$

CRYSTALS-KYBER, Saber

CRYSTALS-KYBER and SABER

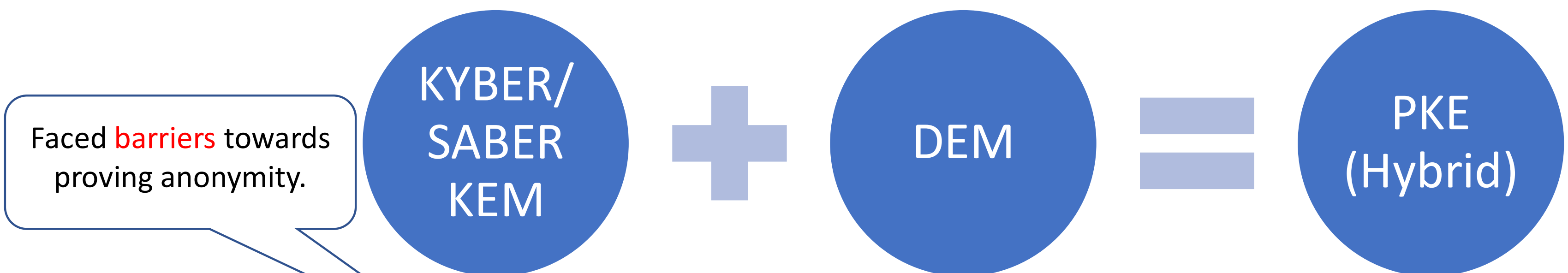
Public-Key Encryption/KEMs

- Classic McEliece
- CRYSTALS-KYBER
- NTRU
- SABER

Public-Key Encryption/KEMs

- BIKE
- FrodoKEM
- HQC
- NTRU Prime
- SIKE

$KEM = (KGen, Encap, Decap)$ $\quad DEM = (Enc^{sym}, Dec^{sym})$ $\quad PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + γ-spread base PKE

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

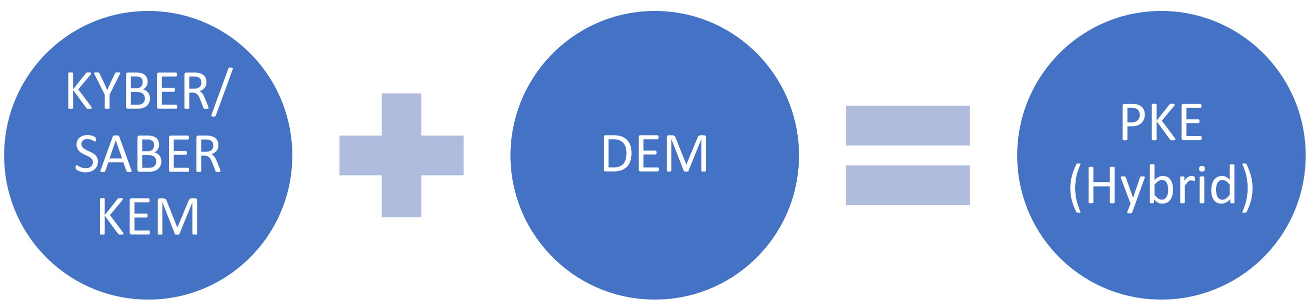
$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure

CRYSTALS-KYBER and SABER

Public-Key Encryption/KEMs	Public-Key Encryption/KEMs
Classic McEliece	BIKE
CRYSTALS-KYBER	FrodoKEM
NTRU	HQC
SABER	NTRU Prime
	SIKE

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



Security analysis of $FO^{\not\perp}$ (e.g., by Jiang et. al.) should not directly apply!

$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + γ-spread base PKE

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure

CRYSTALS-KYBER and SABER

Public-Key Encryption/KEMs

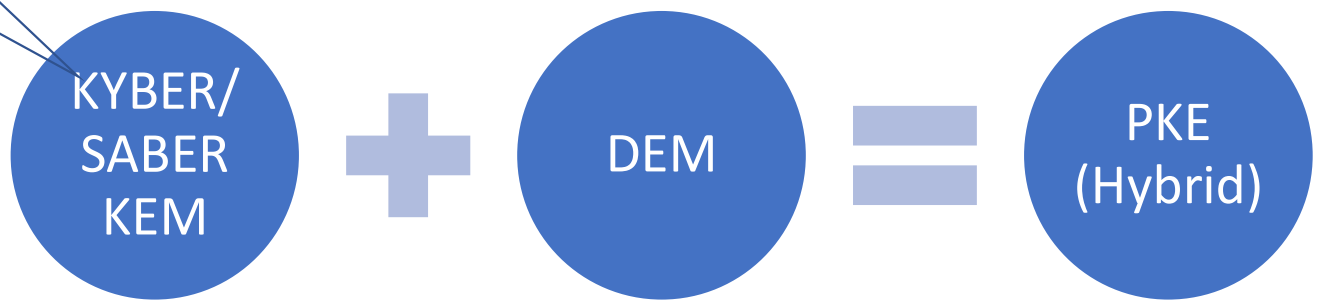
- Classic McEliece
- CRYSTALS-KYBER
- NTRU
- SABER

Public-Key Encryption/KEMs

- BIKE
- FrodoKEM
- HQC
- NTRU Prime
- SIKE

Is strongly "robust". [Grubbs-Maram-Paterson @Eurocrypt'22]

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + γ-spread base PKE

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure

CRYSTALS-KYBER and SABER

<u>Public-Key Encryption/KEMs</u>

- Classic McEliece
- CRYSTALS-KYBER
- NTRU
- SABER

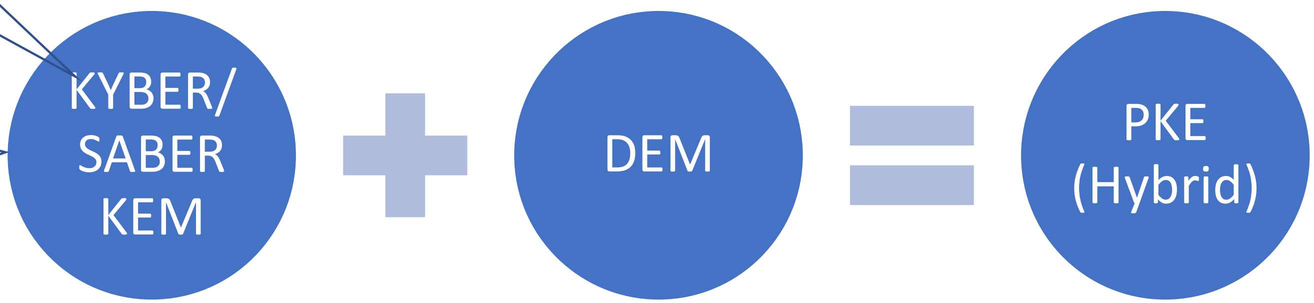
<u>Public-Key Encryption/KEMs</u>

- BIKE
- FrodoKEM
- HQC
- NTRU Prime
- SIKE

Is strongly "robust". [Grubbs-Maram-Paterson @Eurocrypt'22]

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$

$Decap(sk_{Bob}, c) \neq Decap(sk_{Dave}, c)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + γ-spread base PKE

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure

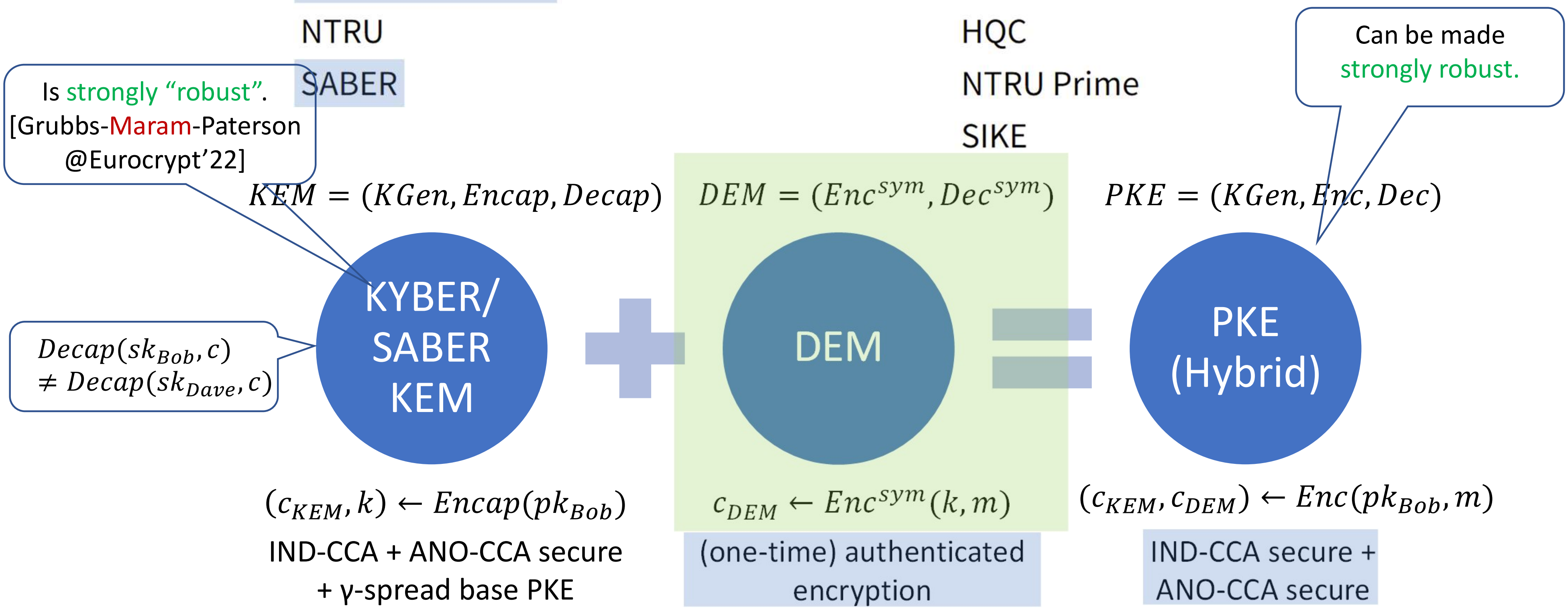
CRYSTALS-KYBER and SABER

<u>Public-Key Encryption/KEMs</u>

Classic McEliece
CRYSTALS-KYBER
NTRU
SABER

<u>Public-Key Encryption/KEMs</u>

BIKE
FrodoKEM
HQC
NTRU Prime
SIKE



FrodoKEM

<u>Public-Key Encryption/KEMs</u>

Classic McEliece

CRYSTALS-KYBER

NTRU

SABER

<u>Public-Key Encryption/KEMs</u>

BIKE

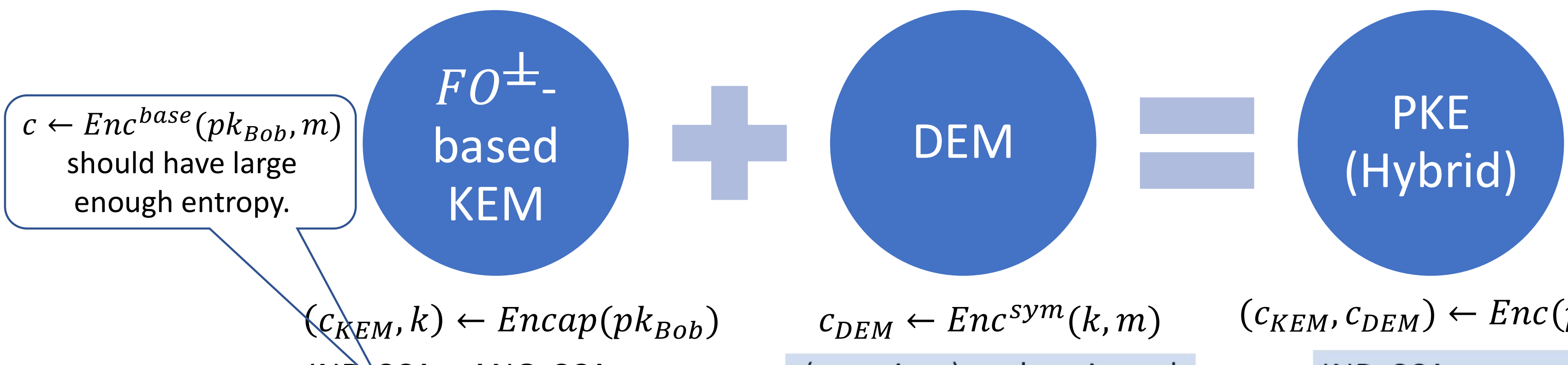
FrodoKEM

HQC

NTRU Prime

SIKE

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + γ-spread base PKE

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure

FrodoKEM

Public-Key Encryption/KEMs	Public-Key Encryption/KEMs
Classic McEliece	BIKE
CRYSTALS-KYBER	FrodoKEM
NTRU	HQC
SABER	NTRU Prime
	SIKE

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$

Frodo KEM + DEM = PKE (Hybrid)

$c \leftarrow Enc^{base}(pk_{Bob}, m)$ should have large enough entropy.

$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + γ-spread base PKE

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

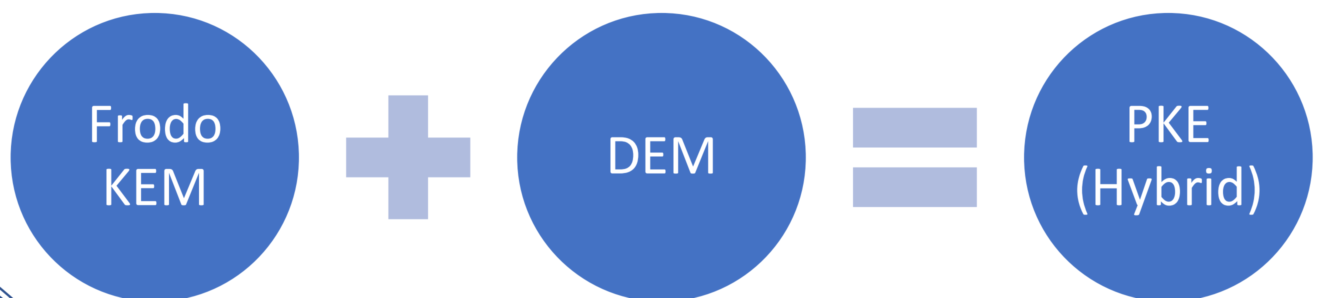
$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure

FrodoKEM

<u>Public-Key Encryption/KEMs</u>	<u>Public-Key Encryption/KEMs</u>
Classic McEliece	BIKE
CRYSTALS-KYBER	FrodoKEM
NTRU	HQC
SABER	NTRU Prime
	SIKE

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



?

$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + γ-spread base PKE

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure

FrodoKEM

Public-Key Encryption/KEMs	Public-Key Encryption/KEMs
Classic McEliece	BIKE
CRYSTALS-KYBER	FrodoKEM
NTRU	HQC
SABER	NTRU Prime
	SIKE

KGen'	$\mathsf{Encap}(\mathsf{pk})$	$\mathsf{Decap}(\mathsf{sk}', c)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$	$m \leftarrow_{\$} \mathcal{M}$	Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
$s \leftarrow_{\$} \mathcal{M}$	$r \leftarrow G(m)$	$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
$\mathsf{sk}' = (\mathsf{sk}, s)$	$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	$r' \leftarrow G(m')$
return $(\mathsf{pk}, \mathsf{sk}')$	$k \leftarrow H(m, c)$	$c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	return (c, k)	**if** $c' = c$ **then**
		return $H(m', c)$
		else return $H(s, c)$

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

KGen'	$\mathsf{Encap}(\mathsf{pk})$	$\mathsf{Decap}(\mathsf{sk}', c)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$	$m \leftarrow_{\$} \mathcal{M}$	Parse $\mathsf{sk}' = (\mathsf{sk}, \mathsf{pk}, F(\mathsf{pk}), s)$
$s \leftarrow_{\$} \mathcal{M}$	$(\hat{k}, r) \leftarrow G(F(pk), m)$	$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
$\mathsf{sk}' \leftarrow (\mathsf{sk}, \mathsf{pk}, F(\mathsf{pk}), s)$	$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	$(\hat{k}', r') \leftarrow G(F(\mathsf{pk}), m')$
return $(\mathsf{pk}, \mathsf{sk}')$	$k \leftarrow H(\hat{k}, c)$	$c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	return (c, k)	**if** $c' = c$ **then**
		return $H(\hat{k}', c)$
		else return $H(s, c)$

FrodoKEM

FrodoKEM

Public-Key Encryption/KEMs	Public-Key Encryption/KEMs
Classic McEliece	BIKE
CRYSTALS-KYBER	FrodoKEM
NTRU	HQC
SABER	NTRU Prime
	SIKE

"$k \leftarrow H(m, c)$"

KGen'	$\mathsf{Encap}(\mathsf{pk})$	$\mathsf{Decap}(\mathsf{sk}', c)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$	$m \leftarrow\!\!\$\ \mathcal{M}$	Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
$s \leftarrow\!\!\$\ \mathcal{M}$	$r \leftarrow G(m)$	$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
$\mathsf{sk}' = (\mathsf{sk}, s)$	$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	$r' \leftarrow G(m')$
return $(\mathsf{pk}, \mathsf{sk}')$	$k \leftarrow H(m, c)$	$c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	return (c, k)	**if** $c' = c$ **then**
		return $H(m', c)$
		else return $H(s, c)$

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

"$k \leftarrow H(G(m), c)$"

KGen'	$\mathsf{Encap}(\mathsf{pk})$	$\mathsf{Decap}(\mathsf{sk}', c)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$	$m \leftarrow\!\!\$\ \mathcal{M}$	Parse $\mathsf{sk}' = (\mathsf{sk}, \mathsf{pk}, F(\mathsf{pk}), s)$
$s \leftarrow\!\!\$\ \mathcal{M}$	$(\hat{k}, r) \leftarrow G(F(pk), m)$	$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
$\mathsf{sk}' \leftarrow (\mathsf{sk}, \mathsf{pk}, F(\mathsf{pk}), s)$	$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	$(\hat{k}', r') \leftarrow G(F(\mathsf{pk}), m')$
return $(\mathsf{pk}, \mathsf{sk}')$	$k \leftarrow H(\hat{k}, c)$	$c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	return (c, k)	**if** $c' = c$ **then**
		return $H(\hat{k}', c)$
		else return $H(s, c)$

FrodoKEM

FrodoKEM

Public-Key Encryption/KEMs

Classic McEliece
CRYSTALS-KYBER
NTRU
SABER

Public-Key Encryption/KEMs

BIKE
FrodoKEM
HQC
NTRU Prime
SIKE

Only nested hashing of m and not c.

“$k \leftarrow H(m, c)$”

KGen′	Encap(pk)	Decap(sk′, c)
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$	$m \leftarrow_\$ \mathcal{M}$	Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
$s \leftarrow_\$ \mathcal{M}$	$r \leftarrow G(m)$	$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
$\mathsf{sk}' = (\mathsf{sk}, s)$	$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	$r' \leftarrow G(m')$
return $(\mathsf{pk}, \mathsf{sk}')$	$k \leftarrow H(m, c)$	$c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	return (c, k)	**if** $c' = c$ **then**
		return $H(m', c)$
		else return $H(s, c)$

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

“$k \leftarrow H(G(m), c)$”

KGen′	Encap(pk)	Decap(sk′, c)
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$	$m \leftarrow_\$ \mathcal{M}$	Parse $\mathsf{sk}' = (\mathsf{sk}, \mathsf{pk}, F(\mathsf{pk}), s)$
$s \leftarrow_\$ \mathcal{M}$	$(\hat{k}, r) \leftarrow G(F(\mathsf{pk}), m)$	$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
$\mathsf{sk}' \leftarrow (\mathsf{sk}, \mathsf{pk}, F(\mathsf{pk}), s)$	$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	$(\hat{k}', r') \leftarrow G(F(\mathsf{pk}), m')$
return $(\mathsf{pk}, \mathsf{sk}')$	$k \leftarrow H(\hat{k}, c)$	$c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	return (c, k)	**if** $c' = c$ **then**
		return $H(\hat{k}', c)$
		else return $H(s, c)$

FrodoKEM

FrodoKEM

Public-Key Encryption/KEMs	Public-Key Encryption/KEMs
Classic McEliece	BIKE
CRYSTALS-KYBER	FrodoKEM
NTRU	HQC
SABER	NTRU Prime
	SIKE

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$

Frodo KEM + DEM = PKE (Hybrid)

Security analysis of $FO^{\not\perp}$ (e.g., by Jiang et. al.) should not directly apply!

$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + γ-spread base PKE

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

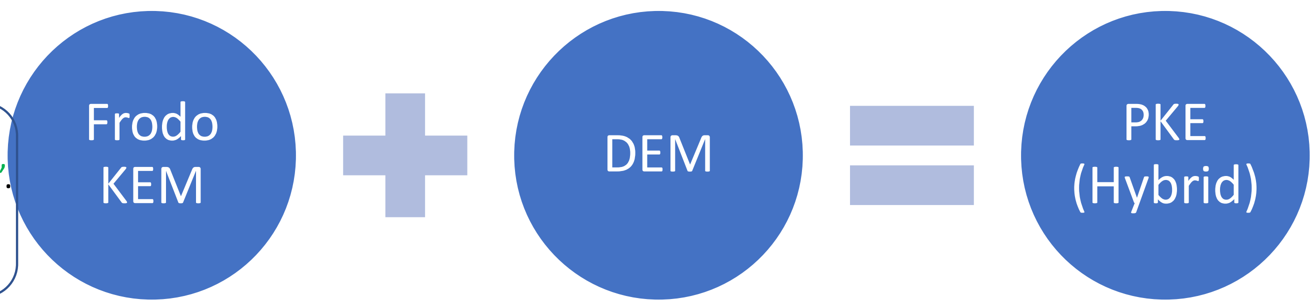
$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure

FrodoKEM

<u>Public-Key Encryption/KEMs</u>	<u>Public-Key Encryption/KEMs</u>
Classic McEliece	BIKE
CRYSTALS-KYBER	FrodoKEM
NTRU	HQC
SABER	NTRU Prime
	SIKE

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$



But (provable) IND-CCA security can be "recovered". [Grubbs-Maram-Paterson @Eurocrypt'22]

$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure + γ-spread base PKE

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure + ANO-CCA secure

FrodoKEM

Public-Key Encryption/KEMs

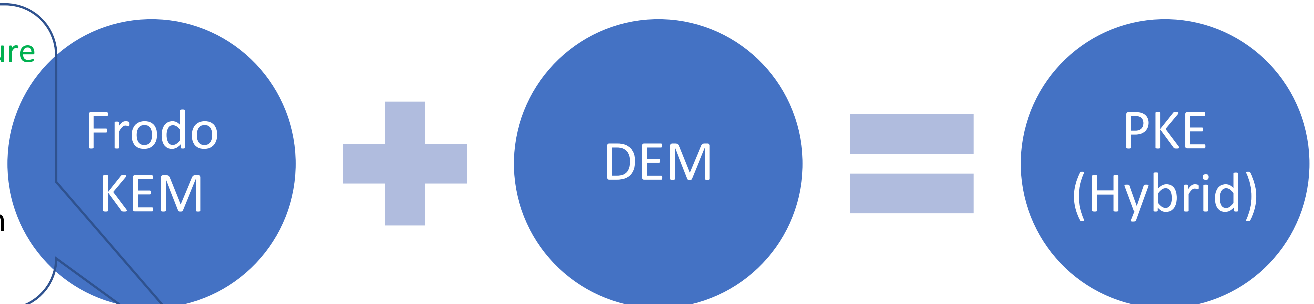
- Classic McEliece
- CRYSTALS-KYBER
- NTRU
- SABER

Public-Key Encryption/KEMs

- BIKE
- FrodoKEM
- HQC
- NTRU Prime
- SIKE

$KEM = (KGen, Encap, Decap)$ $DEM = (Enc^{sym}, Dec^{sym})$ $PKE = (KGen, Enc, Dec)$

FrodoKEM is ANO-CCA secure
and
strongly "robust"
in the QROM.
[Grubbs-Maram-Paterson
@Eurocrypt'22]



$(c_{KEM}, k) \leftarrow Encap(pk_{Bob})$

IND-CCA + ANO-CCA secure
+ γ-spread base PKE

$c_{DEM} \leftarrow Enc^{sym}(k, m)$

(one-time) authenticated encryption

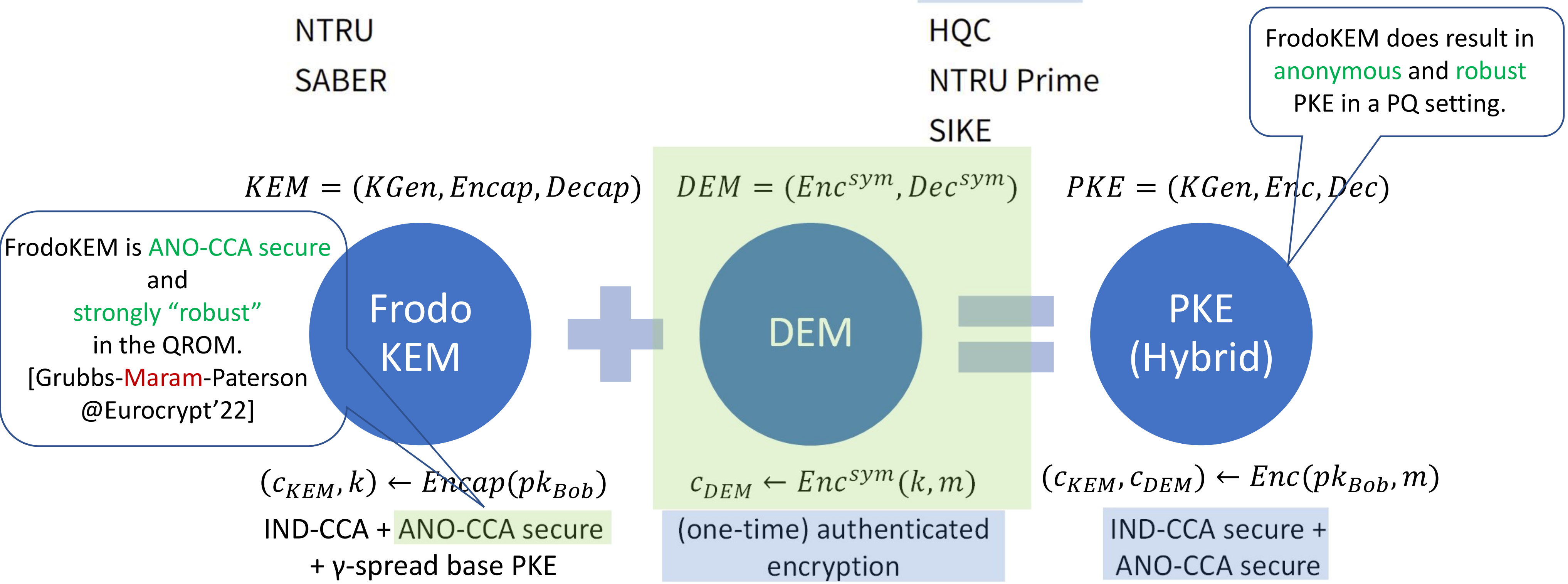
$(c_{KEM}, c_{DEM}) \leftarrow Enc(pk_{Bob}, m)$

IND-CCA secure +
ANO-CCA secure

FrodoKEM

Public-Key Encryption/KEMs
- Classic McEliece
- CRYSTALS-KYBER
- NTRU
- SABER

Public-Key Encryption/KEMs
- BIKE
- FrodoKEM
- HQC
- NTRU Prime
- SIKE



FrodoKEM

Federal Office
for Information Security

BSI – Technical Guideline

Designation:	**Cryptographic Mechanisms: Recommendations and Key Lengths**
Abbreviation:	BSI TR-02102-1
Version:	2023-01
As of:	January 9, 2023

mceliece6688128f and mceliece8192128f [3, Section 7] are assessed to be cryptographically suitable to protect confidential information on a long-term basis at the security level aimed at in this Technical Guideline. This is a very conservative assessment that includes a significant margin of security with respect to future cryptanalytic advances. It is possible that in future revisions of this guideline other parameter choices and PQC mechanisms may also be deemed technically suitable.

FrodoKEM will not be standardised as part of NIST's PQC project. This is mainly due to considerations of the efficiency of the mechanism, there are currently no doubts about its security [2]. Classic McEliece was included in the fourth round of the NIST project and could possibly be standardised at the end of the project. The BSI therefore maintains the recommendation of FrodoKEM and Classic McEliece as PQC mechanisms with a high security margin against future attacks. More details can be found in the BSI-guide "Quantum-safe cryptography" [37].

FrodoKEM

Federal Office
for Information Security

BSI – Technical Guideline

Designation: **Cryptographic Mechanisms: Recommendations and Key Lengths**

Abbreviation: BSI TR-02102-1

Version: 2023-01

As of: January 9, 2023

Technical Guideline – Cryptographic Algorithms and Key Lengths

mceliece6688128f and mceliece8192128f [3, Section 7] are assessed to be cryptographically suitable to protect confidential information on a long-term basis at the security level aimed at in this Technical Guideline. This is a very conservative assessment that includes a significant margin of security with respect to future cryptanalytic advances. It is possible that in future revisions of this guideline other parameter choices and PQC mechanisms may also be deemed technically suitable.

FrodoKEM will not be standardised as part of NIST's PQC project. This is mainly due to considerations of the efficiency of the mechanism, there are currently no doubts about its security [2]. Classic McEliece was included in the fourth round of the NIST project and could possibly be standardised at the end of the project. The BSI therefore maintains the recommendation of FrodoKEM and Classic McEliece as PQC mechanisms with a high security margin against future attacks. More details can be found in the BSI-guide "Quantum-safe cryptography" [37].

Other Contributions

Other Contributions

$\mathsf{Encap}(\mathsf{pk})$	$\mathsf{Decap}(\mathsf{sk}, c)$
$m \leftarrow_{\$} \mathcal{M}$	Parse $c = (c_1, c_2)$
$c_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m; G(m))$	$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c_1)$
$c_2 \leftarrow H'(m)$	$c'_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; G(m'))$
	if $c'_1 = c_1 \wedge H'(m') = c_2$ **then**
$c \leftarrow (c_1, c_2)$	
$k = H(m, c)$	**return** $H(m', c)$
return (c, k)	**else return** $\perp$

$\mathsf{HFO}^{\perp}$

Other Contributions

Encap(pk)	Decap(sk, c)
$m \leftarrow_{\$} \mathcal{M}$	Parse $c = (c_1, c_2)$
$c_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m; G(m))$	$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c_1)$
$c_2 \leftarrow H'(m)$	$c_1' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; G(m'))$
	$\textbf{if } c_1' = c_1 \wedge H'(m') = c_2 \textbf{ then}$
$c \leftarrow (c_1, c_2)$	
$k = H(m, c)$	$\quad \textbf{return } H(m', c)$
$\textbf{return } (c, k)$	$\textbf{else return } \perp$

$\mathsf{HFO}^{\perp}$

Results in IND-CCA secure KEMs in the QROM. [Jiang-Zhang-Ma@PKC'19]

Other Contributions

$\mathsf{Encap}(\mathsf{pk})$	$\mathsf{Decap}(\mathsf{sk}, c)$
$m \leftarrow_{\$} \mathcal{M}$	Parse $c = (c_1, c_2)$
$c_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m; G(m))$	$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c_1)$
$c_2 \leftarrow H'(m)$	$c_1' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; G(m'))$
$\boxed{c_2 \leftarrow H'(m, c_1)}$	**if** $c_1' = c_1 \wedge H'(m') = c_2$ **then**
$c \leftarrow (c_1, c_2)$	$\boxed{\textbf{if } c_1' = c_1 \wedge H'(m', c_1) = c_2 \textbf{ then}}$
$k = H(m, c)$	**return** $H(m', c)$
return (c, k)	**else return** $\perp$

$\mathsf{HFO}^{\perp}$ $\boxed{\mathsf{HFO}^{\perp'}}$

Results in IND-CCA secure KEMs in the QROM. [Jiang-Zhang-Ma@PKC'19]

Other Contributions

Encap(pk)	Decap(sk, c)
$m \leftarrow_\$ \mathcal{M}$	Parse $c = (c_1, c_2)$
$c_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m; G(m))$	$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c_1)$
$c_2 \leftarrow H'(m)$	$c_1' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; G(m'))$
$\boxed{c_2 \leftarrow H'(m, c_1)}$	**if** $c_1' = c_1 \wedge H'(m') = c_2$ **then**
$c \leftarrow (c_1, c_2)$	$\boxed{\textbf{if } c_1' = c_1 \wedge H'(m', c_1) = c_2 \textbf{ then}}$
$k = H(m, c)$	**return** $H(m', c)$
return (c, k)	**else return** $\perp$

$\mathsf{HFO}^{\perp}$ $\boxed{\mathsf{HFO}^{\perp'}}$

Results in IND-CCA secure KEMs in the QROM. [Jiang-Zhang-Ma@PKC'19]

Results in IND-CCA, ANO-CCA and SROB secure KEMs in the QROM. [Grubbs-Maram-Paterson @Eurocrypt'22]

Summary

Summary

- We provide insights into obtaining anonymous and robust hybrid PKE schemes – via the KEM-DEM composition – when the KEM is implicitly rejecting (i.e., non-robust).

Summary

- We provide insights into obtaining anonymous and robust hybrid PKE schemes – via the KEM-DEM composition – when the KEM is <u>implicitly rejecting</u> (i.e., non-robust).
- We showed that the $\mathsf{FO}^{\not\perp}$ transform does result in ANO-CCA secure and "robust" KEMs in a <u>post-quantum setting</u> (i.e., the QROM).

Summary

- We provide insights into obtaining anonymous and robust hybrid PKE schemes – via the KEM-DEM composition – when the KEM is implicitly rejecting (i.e., non-robust).
- We showed that the $\mathsf{FO}^{\not\perp}$ transform does result in ANO-CCA secure and "robust" KEMs in a post-quantum setting (i.e., the QROM).
- Hybrid PKE schemes derived from Classic McEliece cannot be (strongly) robust.
    - Though they can be made ANO-CCA secure as shown in [Xagawa@Eurocrypt'22].

Summary

- We provide insights into obtaining anonymous and robust hybrid PKE schemes – via the KEM-DEM composition – when the KEM is implicitly rejecting (i.e., non-robust).
- We showed that the $\mathsf{FO}^{\not\perp}$ transform does result in ANO-CCA secure and "robust" KEMs in a post-quantum setting (i.e., the QROM).
- Hybrid PKE schemes derived from Classic McEliece cannot be (strongly) robust.
  - Though they can be made ANO-CCA secure as shown in [Xagawa@Eurocrypt'22].
- We identified barriers towards proving IND-CCA and ANO-CCA security of CRYSTALS-KYBER and SABER in the QROM.
  - At the same time, we showed they do result in strongly robust hybrid PKE schemes.

Summary

- We provide insights into obtaining anonymous and robust hybrid PKE schemes – via the KEM-DEM composition – when the KEM is implicitly rejecting (i.e., non-robust).
- We showed that the $\mathsf{FO}^{\not\perp}$ transform does result in ANO-CCA secure and "robust" KEMs in a post-quantum setting (i.e., the QROM).
- Hybrid PKE schemes derived from Classic McEliece cannot be (strongly) robust.
  - Though they can be made ANO-CCA secure as shown in [Xagawa@Eurocrypt'22].
- We identified barriers towards proving IND-CCA and ANO-CCA security of CRYSTALS-KYBER and SABER in the QROM.
  - At the same time, we showed they do result in strongly robust hybrid PKE schemes.
- Finally, we showed that FrodoKEM does result in ANO-CCA secure and strongly robust hybrid PKE schemes in the QROM.

Recent Developments

Recent Developments

NIST Announces First Four Quantum-Resistant Cryptographic Algorithms

Federal agency reveals the first group of winners from its six-year competition.

July 05, 2022

For general encryption, used when we access secure websites, NIST has selected the CRYSTALS-Kyber algorithm. Among its advantages are comparatively small encryption keys that two parties can exchange easily, as well as its speed of operation.

Recent Developments

NIST Announces First Four Quantum-Resistant Cryptographic Algorithms

Federal agency reveals the first group of winners from its six-year competition.

July 05, 2022

For general encryption, used when we access secure websites, NIST has selected the CRYSTALS-Kyber algorithm. Among its advantages are comparatively small encryption keys that two parties can exchange easily, as well as its speed of operation.

Provable IND-CCA security in the QROM unclear.
[Grubbs-Maram-Paterson @Eurocrypt'22]

Recent Developments

NIST Announces First Four Quantum-Resistant Cryptographic Algorithms

Federal agency reveals the first group of winners from its six-year competition.

July 05, 2022

For general encryption, used when we access secure websites, NIST has selected the CRYSTALS-Kyber algorithm. Among its advantages are comparatively small encryption keys that two parties can exchange easily, as well as its speed of operation.

Provable IND-CCA security in the QROM unclear.
[Grubbs-Maram-Paterson @Eurocrypt'22]

Name	KEM IND	KEM SPR	KEM ANO	KEM CF	KEM ROB	Hybrid PKE ANO	Hybrid PKE ROB	
Classic McEliece [ABC⁺20]	Y	<u>Y</u>	<u>Y</u>	N	N	<u>Y</u>	N	Section K
Kyber [SAB⁺20]	?	?	?	?	N	?	?	Section L
NTRU [CDH⁺20]	Y	<u>Y</u>	<u>Y</u>	<u>Y</u>	N	<u>Y</u>	<u>Y</u>	Section 5
Saber [DKR⁺20]	?	?	?	?	N	?	?	Section M

[Xagawa@Eurocrypt'22]

Recent Developments

Discussion about Kyber's tweaked FO transform 148 views

Peter Schwabe
to pqc-forum

Dear all,

At the fourth NIST PQC Standardization Workshop we sketched a few possible changes to Kyber that could be considered in the standardization phase; we followed up on those in two e-mails with subjects "Kyber decisions, part 1: symmetric crypto" and "Kyber decisions, part 2: FO transform". The points we brought up for discussion in the first e-mail received quite some feedback and eventually NIST decided to not integrate any of the changes. The second mail received way fewer replies, but Markku asked us for a more concrete description of the proposed change. Apologies that this request remained unanswered for so long! In this mail we would like to follow up and make the suggested change more concrete.

Currently, Kyber's tweaked FO transform looks as follows:

```
Encaps(pk):
(K,r) <- G(m,H(pk))
c <- Encrypt(m,pk,r)
K' <- KDF(K,H(c))
return K',c


Decaps(SK=(sk,pk,z),c):
m' <- Decrypt(sk,c)
(K,r) <- G(m',H(pk))
c' <- Encrypt(m',pk,r)
if(c' == c)
K' <- KDF(K,H(c))
else
K' <- KDF(z,H(c))
return K'
```

The concrete proposal would be to change this to:

```
Encaps(pk):
(K,r) <- G(m,H(pk))
c <- Encrypt(m,pk,r)
return K,c

Decaps(SK=(sk,pk,z),c):
m' <- Decrypt(sk,c)
(K,r) <- G(m',H(pk))
c' <- Encrypt(m',pk,r)
(K',-) <- G(z,c)
if(c' != c)
K <- K'
return K
```

Note that this is the standard FO transform with implicit rejection, except that the hash of the public key is fed as an additional argument into G to derive (K, r). As a reminder, this provides some protection against multi-target decryption-failure attacks and makes Kyber "contributory", i.e., ensures that the shared key depends on high-entropy input from both parties.

The advantages of this change would be the following:

* Encaps avoids hashing over the ciphertext. In our AVX2 optimized implementation this translates to a speedup of ~17%. Note that the speedup on most other platforms and for masked implementations is going to be smaller than that.

* More importantly, this change simplifies proofs and leads to better bounds without requiring a new failure-bound analysis. More specifically, the only direct proofs of the FO originally used by Kyber that we could come up with produces a bound with an additive C(q + q_dec + 1)^3/2^{256} term where C is some constant, q is the number of the adversary queries to the random oracle, and q_dec is the number of the adversaries decryption queries. This is caused by having to deal with collisions in H (when computing H(c)). Alternative proofs via explicit rejection either lead to a worse bound or require to analyze the failure bound in the extractable QROM, which has not been done so far.

Recent Developments

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

Discussion about Kyber's tweaked FO transform 148 views

Peter Schwabe
to pqc-forum

Dear all,

At the fourth NIST PQC Standardization Workshop we sketched a few possible changes to Kyber that could be considered in the standardization phase; we followed up on those in two e-mails with subjects "Kyber decisions, part 1: symmetric crypto" and "Kyber decisions, part 2: FO transform". The points we brought up for discussion in the first e-mail received quite some feedback and eventually NIST decided to not integrate any of the changes. The second mail received way fewer replies, but Markku asked us for a more concrete description of the proposed change. Apologies that this request remained unanswered for so long! In this mail we would like to follow up and make the suggested change more concrete.

Currently, Kyber's tweaked FO transform looks as follows:

```
Encaps(pk):
(K,r) <- G(m,H(pk))
c <- Encrypt(m,pk,r)
K' <- KDF(K,H(c))
return K',c


Decaps(SK=(sk,pk,z),c):
m' <- Decrypt(sk,c)
(K,r) <- G(m',H(pk))
c' <- Encrypt(m',pk,r)
if(c' == c)
K' <- KDF(K,H(c))
else
K' <- KDF(z,H(c))
return K'
```

The concrete proposal would be to change this to:

```
Encaps(pk):
(K,r) <- G(m,H(pk))
c <- Encrypt(m,pk,r)
return K,c

Decaps(SK=(sk,pk,z),c):
m' <- Decrypt(sk,c)
(K,r) <- G(m',H(pk))
c' <- Encrypt(m',pk,r)
(K',-) <- G(z,c)
if(c' != c)
K <- K'
return K
```

Note that this is the standard FO transform with implicit rejection, except that the hash of the public key is fed as an additional argument into G to derive (K, r). As a reminder, this provides some protection against multi-target decryption-failure attacks and makes Kyber "contributory", i.e., ensures that the shared key depends on high-entropy input from both parties.

The advantages of this change would be the following:

* Encaps avoids hashing over the ciphertext. In our AVX2 optimized implementation this translates to a speedup of ~17%. Note that the speedup on most other platforms and for masked implementations is going to be smaller than that.

* More importantly, this change simplifies proofs and leads to better bounds without requiring a new failure-bound analysis. More specifically, the only direct proofs of the FO originally used by Kyber that we could come up with produces a bound with an additive C(q + q_dec + 1)^3/2^{256} term where C is some constant, q is the number of the adversary queries to the random oracle, and q_dec is the number of the adversaries decryption queries. This is caused by having to deal with collisions in H (when computing H(c)). Alternative proofs via explicit rejection either lead to a worse bound or require to analyze the failure bound in the extractable QROM, which has not been done so far.

Recent Developments

Essentially the same as $\mathsf{FO}^{\not\perp}_{m}$ [Hofheinz-Hövelmanns-Kiltz @TCC'17]

Discussion about Kyber's tweaked FO transform 148 views

Peter Schwabe
to pqc-forum

Dear all,

At the fourth NIST PQC Standardization Workshop we sketched a few possible changes to Kyber that could be considered in the standardization phase; we followed up on those in two e-mails with subjects "Kyber decisions, part 1: symmetric crypto" and "Kyber decisions, part 2: FO transform". The points we brought up for discussion in the first e-mail received quite some feedback and eventually NIST decided to not integrate any of the changes. The second mail received way fewer replies, but Markku asked us for a more concrete description of the proposed change. Apologies that this request remained unanswered for so long! In this mail we would like to follow up and make the suggested change more concrete.

Currently, Kyber's tweaked FO transform looks as follows:

```
Encaps(pk):
(K,r) <- G(m,H(pk))
c <- Encrypt(m,pk,r)
K' <- KDF(K,H(c))
return K',c


Decaps(SK=(sk,pk,z),c):
m' <- Decrypt(sk,c)
(K,r) <- G(m',H(pk))
c' <- Encrypt(m',pk,r)
if(c' == c)
K' <- KDF(K,H(c))
else
K' <- KDF(z,H(c))
return K'
```

The concrete proposal would be to change this to:

```
Encaps(pk):
(K,r) <- G(m,H(pk))
c <- Encrypt(m,pk,r)
return K,c

Decaps(SK=(sk,pk,z),c):
m' <- Decrypt(sk,c)
(K,r) <- G(m',H(pk))
c' <- Encrypt(m',pk,r)
(K',-) <- G(z,c)
if(c' != c)
K <- K'
return K
```

Note that this is the standard FO transform with implicit rejection, except that the hash of the public key is fed as an additional argument into G to derive (K, r). As a reminder, this provides some protection against multi-target decryption-failure attacks and makes Kyber "contributory", i.e., ensures that the shared key depends on high-entropy input from both parties.

The advantages of this change would be the following:

* Encaps avoids hashing over the ciphertext. In our AVX2 optimized implementation this translates to a speedup of ~17%. Note that the speedup on most other platforms and for masked implementations is going to be smaller than that.

* More importantly, this change simplifies proofs and leads to better bounds without requiring a new failure-bound analysis. More specifically, the only direct proofs of the FO originally used by Kyber that we could come up with produces a bound with an additive C(q + q_dec + 1)^3/2^{256} term where C is some constant, q is the number of the adversary queries to the random oracle, and q_dec is the number of the adversaries decryption queries. This is caused by having to deal with collisions in H (when computing H(c)). Alternative proofs via explicit rejection either lead to a worse bound or require to analyze the failure bound in the extractable QROM, which has not been done so far.

Recent Developments

Discussion about Kyber's tweaked FO transform

148 views

Peter Schwabe

to pqc-forum

Dear all,

At the fourth NIST PQC Standardization Workshop we sketched a few possible changes to Kyber that could be considered in the standardization phase; we followed up on those in two e-mails with subjects "Kyber decisions, part 1: symmetric crypto" and "Kyber decisions, part 2: FO transform". The points we brought up for discussion in the first e-mail received quite some feedback and eventually NIST decided to not integrate any of the changes. The second mail received way fewer replies, but Markku asked us for a more concrete description of the proposed change. Apologies that this request remained unanswered for so long! In this mail we would like to follow up and make the suggested change more concrete.

Currently, Kyber's tweaked FO transform looks as follows:

```
Encaps(pk):
(K,r) <- G(m,H(pk))
c <- Encrypt(m,pk,r)
K' <- KDF(K,H(c))
return K',c


Decaps(SK=(sk,pk,z),c):
m' <- Decrypt(sk,c)
(K,r) <- G(m',H(pk))
c' <- Encrypt(m',pk,r)
if(c' == c)
K' <- KDF(K,H(c))
else
K' <- KDF(z,H(c))
return K'
```

The concrete proposal would be to change this to:

```
Encaps(pk):
(K,r) <- G(m,H(pk))
c <- Encrypt(m,pk,r)
return K,c

Decaps(SK=(sk,pk,z),c):
m' <- Decrypt(sk,c)
(K,r) <- G(m',H(pk))
c' <- Encrypt(m',pk,r)
(K',-) <- G(z,c)
if(c' != c)
K <- K'
return K
```

Note that this is the standard FO transform with implicit rejection, except that the hash of the public key is fed as an additional argument into G to derive (K, r). As a reminder, this provides some protection against multi-target decryption-failure attacks and makes Kyber "contributory", i.e., ensures that the shared key depends on high-entropy input from both parties.

The advantages of this change would be the following:

* Encaps avoids hashing over the ciphertext. In our AVX2 optimized implementation this translates to a speedup of ~17%. Note that the speedup on most other platforms and for masked implementations is going to be smaller than that.

* More importantly, this change simplifies proofs and leads to better bounds without requiring a new failure-bound analysis. More specifically, the only direct proofs of the FO originally used by Kyber that we could come up with produces a bound with an additive C(q + q_dec + 1)^3/2^{256} term where C is some constant, q is the number of the adversary queries to the random oracle, and q_dec is the number of the adversaries decryption queries. This is caused by having to deal with collisions in H (when computing H(c)). Alternative proofs via explicit rejection either lead to a worse bound or require to analyze the failure bound in the extractable QROM, which has not been done so far.

Recent Developments

Post-Quantum Anonymity of Kyber

Varun Maram[1] and Keita Xagawa[2]

[1] Department of Computer Science, ETH Zurich, Switzerland.
vmaram@inf.ethz.ch
[2] NTT Social Informatics Laboratories, Japan.
keita.xagawa.zv@hco.ntt.co.jp

Recent Developments

Post-Quantum Anonymity of Kyber

Varun Maram[1] and Keita Xagawa[2]

[1] Department of Computer Science, ETH Zurich, Switzerland.
vmaram@inf.ethz.ch
[2] NTT Social Informatics Laboratories, Japan.
keita.xagawa.zv@hco.ntt.co.jp

- Provided concrete proof of IND-CCA security for Kyber (with tweaked FO) in the QROM.

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

Recent Developments

Post-Quantum Anonymity of Kyber

Varun Maram[1] and Keita Xagawa[2]

[1] Department of Computer Science, ETH Zurich, Switzerland.
vmaram@inf.ethz.ch
[2] NTT Social Informatics Laboratories, Japan.
keita.xagawa.zv@hco.ntt.co.jp

- Provided concrete proof of IND-CCA security for Kyber (with tweaked FO) in the QROM.

$$\mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{Kyber}} \leq \mathrm{Adv}^{\mathrm{IND-C}}_{\mathsf{FO}^{\not\perp}_{m}} \quad + \mathrm{Adv}^{\mathrm{CR}}_{F}$$

Recent Developments

KGen′	Encap(pk)	Decap(sk′, c)
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$	$m \leftarrow_{\$} \mathcal{M}$	Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
$s \leftarrow_{\$} \mathcal{M}$	$r \leftarrow G(m)$	$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
$\mathsf{sk}' = (\mathsf{sk}, s)$	$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	$r' \leftarrow G(m')$
return $(\mathsf{pk}, \mathsf{sk}')$	$k \leftarrow H(m, c)$	$c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	return (c, k)	**if** $c' = c$ **then**
		return $H(m', c)$
		else return $H(s, c)$

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

KGen′	Encap(pk)	Decap(sk′, c)
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$	$m \leftarrow_{\$} \mathcal{M}$	Parse $\mathsf{sk}' = (\mathsf{sk}, \mathsf{pk}, F(\mathsf{pk}), s)$
$s \leftarrow_{\$} \mathcal{M}$	$m \leftarrow F(m)$	$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
$\mathsf{sk}' \leftarrow (\mathsf{sk}, \mathsf{pk}, F(\mathsf{pk}), s)$	$(\hat{k}, r) \leftarrow G(F(pk), m)$	$(\hat{k}', r') \leftarrow G(F(\mathsf{pk}), m')$
return $(\mathsf{pk}, \mathsf{sk}')$	$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	$c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	$k \leftarrow \mathsf{KDF}(\hat{k}, F(c))$	**if** $c' = c$ **then**
	return (c, k)	**return** $\mathsf{KDF}(\hat{k}', F(c))$
		else return $\mathsf{KDF}(s, F(c))$

CRYSTALS-KYBER

- Provided concrete proof of IND-CCA security for Kyber (with tweaked FO) in the QROM.

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

“$k \leftarrow H(m, c)$”

“$k \leftarrow H(G(m), F(c))$”

Recent Developments

KGen′	Encap(pk)	Decap(sk′, c)
(pk, sk) ← KGen	$m \leftarrow_{\$} \mathcal{M}$	Parse sk′ = (sk, s)
$s \leftarrow_{\$} \mathcal{M}$	$r \leftarrow G(m)$	$m' \leftarrow$ Dec(sk, c)
sk′ = (sk, s)	$c \leftarrow$ Enc(pk, $m; r$)	$r' \leftarrow G(m')$
return (pk, sk′)	$k \leftarrow H(m, c)$	$c' \leftarrow$ Enc(pk, $m'; r'$)
	return (c, k)	**if** $c' = c$ **then**
		return $H(m', c)$
		else return $H(s, c)$

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

KGen′	Encap(pk)	Decap(sk′, c)
(pk, sk) ← KGen	$m \leftarrow_{\$} \mathcal{M}$	Parse sk′ = (sk, pk, F(pk), s)
$s \leftarrow_{\$} \mathcal{M}$	$m \leftarrow F(m)$	$m' \leftarrow$ Dec(sk, c)
sk′ ← (sk, pk, F(pk), s)	$(\hat{k}, r) \leftarrow G(F(pk), m)$	$(\hat{k}', r') \leftarrow G(F(\mathsf{pk}), m')$
return (pk, sk′)	$c \leftarrow$ Enc(pk, $m; r$)	$c' \leftarrow$ Enc(pk, $m'; r'$)
	$k \leftarrow$ KDF($\hat{k}$, $F(c)$)	**if** $c' = c$ **then**
	return (c, k)	**return** KDF($\hat{k}'$, $F(c)$)
		else return KDF(s, $F(c)$)

CRYSTALS-KYBER

- Provided concrete proof of IND-CCA security for Kyber (with tweaked FO) in the QROM.

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

“$k \leftarrow H(m, c)$”

“$k \leftarrow H(G(m), F(c))$”

Collision-resistance of nested hash F.

Recent Developments

Post-Quantum Anonymity of Kyber

Varun Maram[1] and Keita Xagawa[2]

[1] Department of Computer Science, ETH Zurich, Switzerland.
vmaram@inf.ethz.ch
[2] NTT Social Informatics Laboratories, Japan.
keita.xagawa.zv@hco.ntt.co.jp

- Provided concrete proof of IND-CCA security for Kyber (with tweaked FO) in the QROM.

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

- Showed Kyber also results in ANO-CCA secure and strongly robust hybrid PKE schemes in the QROM.

Recent Developments

Post-Quantum Anonymity of Kyber

Varun Maram[1] and Keita Xagawa[2]

[1] Department of Computer Science, ETH Zurich, Switzerland.
vmaram@inf.ethz.ch
[2] NTT Social Informatics Laboratories, Japan.
keita.xagawa.zv@hco.ntt.co.jp

- Provided concrete proof of IND-CCA security for Kyber (with tweaked FO) in the QROM.

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

- Showed Kyber also results in ANO-CCA secure and strongly robust hybrid PKE schemes in the QROM.
- Work to appear at [PKC'23] (co-winner of the "Best Paper Award").

Future Directions

Future Directions

4.A.2 Security Definition for Encryption/Key-Establishment
NIST intends to standardize one or more schemes that enable "semantically secure" encryption or key encapsulation with respect to adaptive chosen ciphertext attack, for general use. This property is generally denoted *IND-CCA2 security* in academic literature.

Future Directions

4.A.2 Security Definition for Encryption/Key-Establishment
NIST intends to standardize one or more schemes that enable "semantically secure" encryption or key encapsulation with respect to adaptive chosen ciphertext attack, for general use. This property is generally denoted *IND-CCA2 security* in academic literature.

"not sufficient"

- Anonymity and Robustness:
  - [Grubbs-Maram-Paterson @Eurocrypt'22]
  - [Xagawa@Eurocrypt'22]
  - [Maram-Xagawa@PKC'23]

Future Directions

4.A.2 Security Definition for Encryption/Key-Establishment
NIST intends to standardize one or more schemes that enable "semantically secure" encryption or key encapsulation with respect to adaptive chosen ciphertext attack, for general use. This property is generally denoted *IND-CCA2 security* in academic literature.

"not sufficient"

- Anonymity and Robustness:
  - [Grubbs-Maram-Paterson @Eurocrypt'22]
  - [Xagawa@Eurocrypt'22]
  - [Maram-Xagawa@PKC'23]

Secret key "shared" across multiple parties

- Threshold CCA security:
  - [Cong-Cozzo-Maram-Smart @Asiacrypt'21]

Future Directions

4.A.2 Security Definition for Encryption/Key-Establishment
NIST intends to standardize one or more schemes that enable "semantically secure" encryption or key encapsulation with respect to adaptive chosen ciphertext attack, for general use. This property is generally denoted *IND-CCA2 security* in academic literature.

"not sufficient"

- Anonymity and Robustness:
  - [Grubbs-Maram-Paterson @Eurocrypt'22]
  - [Xagawa@Eurocrypt'22]
  - [Maram-Xagawa@PKC'23]
- Threshold CCA security:
  - [Cong-Cozzo-Maram-Smart @Asiacrypt'21]
- Other properties?

Future Directions

4.A.2 Security Definition for Encryption/Key-Establishment
NIST intends to standardize one or more schemes that enable “semantically secure” encryption or key encapsulation with respect to adaptive chosen ciphertext attack, for general use. This property is generally denoted *IND-CCA2 security* in academic literature.

“not sufficient”

- Anonymity and Robustness:
  - [Grubbs-Maram-Paterson @Eurocrypt’22]
  - [Xagawa@Eurocrypt’22]
  - [Maram-Xagawa@PKC’23]
- Threshold CCA security:
  - [Cong-Cozzo-Maram-Smart @Asiacrypt’21]
- Other properties?

“not necessary”

- “One-time” CCA security:
  - [Huguenin-Dumittan-Vaudenay @Eurocrypt’22]
  - [Günther-Maram, Work in Progress]

Future Directions

> **4.A.2 Security Definition for Encryption/Key-Establishment**
> NIST intends to standardize one or more schemes that enable "semantically secure" encryption or key encapsulation with respect to adaptive chosen ciphertext attack, for general use. This property is generally denoted *IND-CCA2 security* in academic literature.

"not sufficient"

- Anonymity and Robustness:
  - [Grubbs-Maram-Paterson @Eurocrypt'22]
  - [Xagawa@Eurocrypt'22]
  - [Maram-Xagawa@PKC'23]
- Threshold CCA security:
  - [Cong-Cozzo-Maram-Smart @Asiacrypt'21]
- Other properties?

"not necessary"

- "One-time" CCA security:
  - [Huguenin-Dumittan-Vaudenay @Eurocrypt'22]
  - [Günther-Maram, Work in Progress]

Secure against adversaries making a single decryption query.