UNIVERSITY OF WATERLOO

Adding functionality to post-quantum cryptography with variants of the Fujisaki–Okamoto transform

Douglas Stebila

University of Waterloo

Based on a series of joint work with Lewis Glabush (C&O MMath → EPFL PhD), Felix Günther (IBM Research Zürich), Britta Hale (Naval Postgraduate School), Kathrin Hövelmanns (TU Eindhoven), and Elizabeth Van Oorschot (C&O URA, McGill).

We acknowledge the support of NSERC.

Some motivating attacks

Public-key encryption

Alice generates a key pair and publishes her public key:

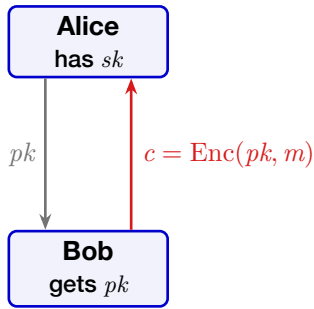
$$\text{KeyGen} \to (pk, sk)$$

Anyone can encrypt a message to Alice:

$$\text{Enc}(pk, m) \to c$$

Only Alice can decrypt a ciphertext:

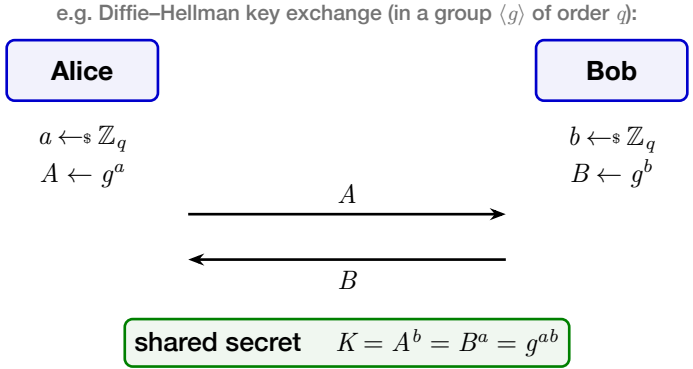
$$\text{Dec}(sk, c) \to m$$



…but public key encryption is a polite fiction

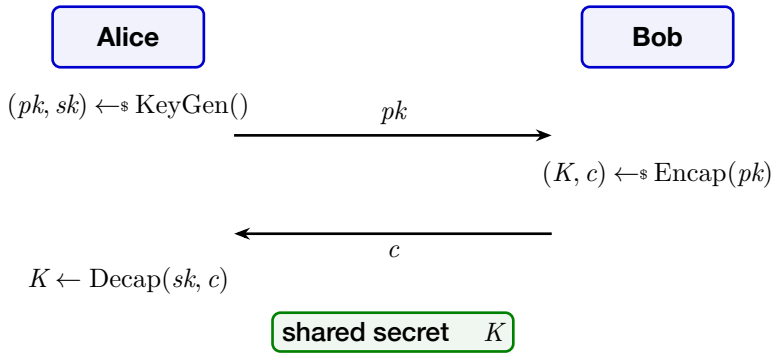
We almost never encrypt application data with public-key encryption.

Instead: use asymmetric algorithms to agree on a **shared secret**, then encrypt with fast symmetric crypto.



Key encapsulation mechanisms (KEMs)

A generalization of Diffie–Hellman key exchange:



KEMs are at the heart of the post-quantum transition

Post-quantum KEM standardization:

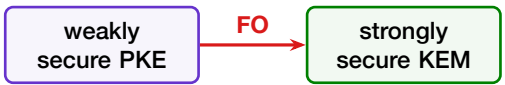
- ML-KEM* — NIST standard, 2024, based on module lattices
- HQC — NIST selected, 2025, based on error-correcting codes
- FrodoKEM* — ISO standard, 2026, based on lattices

Adoption:

- TLS (the web)
- Signal, SSH, iMessage, …

How to build a KEM?

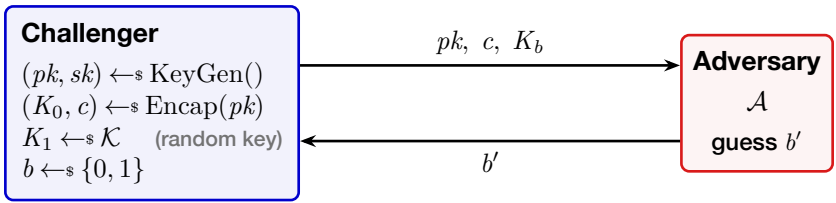
Every standardized post-quantum KEM is built via the **Fujisaki–Okamoto (FO) transform**.



* Both ML-KEM and FrodoKEM have co-authors from UW C&O (John Schanck, D.S.)

What does it mean for a KEM to be secure?

A KEM is secure if its key is **indistinguishable from random**, even to an adversary who sees the public key and ciphertext.



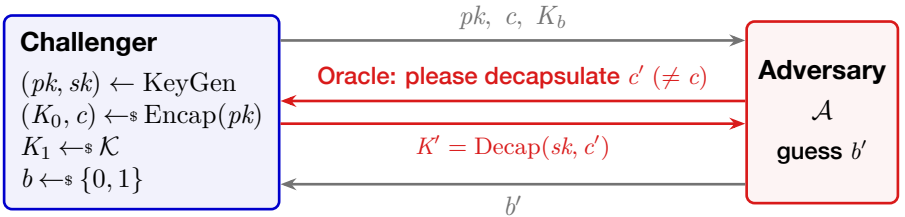
A KEM is **IND-CPA-secure** if the adversary's advantage is small:

$$\mathbf{Adv}^{\mathbf{IND\text{-}CPA}}_{\mathrm{KEM}}(\mathcal{A}) = \left|\mathrm{Prob}[b' = b] - \tfrac{1}{2}\right|$$

CPA = chosen-plaintext attack: the adversary is *passive* — it only sees the public key and the challenge.

Active attackers: IND-CCA (chosen ciphertext attacks)

Real attackers don't just watch — they **inject and modify** ciphertexts and observe what the receiver does. So we give the adversary a decapsulation oracle:



Indistinguishability against chosen ciphertext attacks

Note

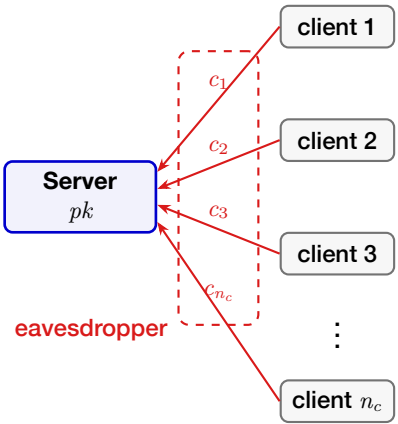
For IND-CCA, adversary is given one challenge ciphertext to break.

Attack 1 — the setting

One server, one key pair, *many* clients.

The adversary passively collects n_c encapsulation ciphertexts off the wire.

Goal: recover the shared key of *any one* of them.



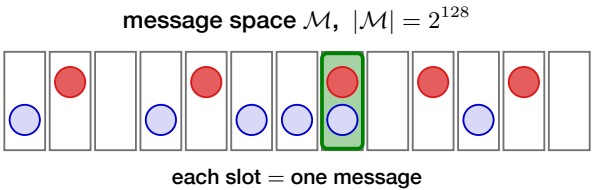
FO makes encryption a function of the message

Inside the FO transform, encryption is **derandomized**:

$$c = \mathrm{Enc}(pk, m;\ G(m))$$ (the randomness of Enc is $G(m)$, a hash of m).

So c is a deterministic, *public* function of m — anyone can recompute it.

$\Longrightarrow$ If the message space is small, an eavesdropper can just **guess and check**.



• challenge ciphertexts • adversary's N guesses collision = a recovered key

Attack 1: success probability

A pure birthday / collision argument:

$$\mathrm{Prob}[\textbf{collision}] \approx \frac{N n_c}{|\mathcal{M}|}.$$

For FrodoKEM-640 & HQC-128:

$$|\mathcal{M}| = 2^{128}$$

Take $N = n_c = 2^{64}$:
high success probability.

Multi-ciphertext security is halved

128-bit single-challenge security
$\longrightarrow \sim 64$-bit **multi-target** security.

Identified by NIST & Bernstein against FrodoKEM-640 and HQC-128.

NIST PQC team, private communication, 2021.
Bernstein. Multi-ciphertext security degradation for lattices, https://eprint.iacr.org/2022/1580, 2022.

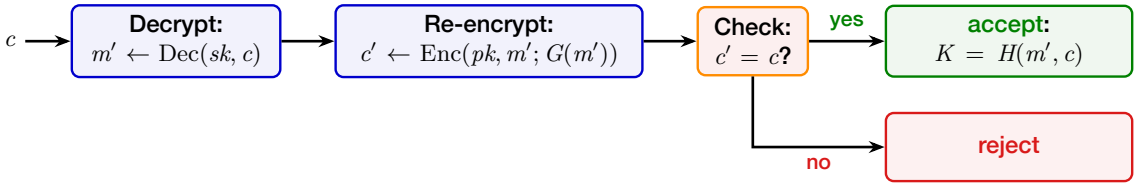
FO decapsulation re-encrypts to check the ciphertext

Attack 2: implementation bug

After derandomization, each message has *exactly one* valid ciphertext:

$$c = \mathrm{Enc}(pk, m;\ G(m)).$$

So on receipt, the FO transform doesn't just decrypt; it **re-encrypts and checks**.



Idea: if the ciphertext was well-formed, the adversary "knew" the message, so decrypting it doesn't help the adversary.

This check is what upgrades *passive* IND-CPA security to *active* IND-CCA security: a malformed ciphertext fails the check and is rejected, so the decapsulation oracle leaks nothing.

Attack 2: a missing re-encryption check

Remove that one check…
…and decapsulation will accept malformed ciphertexts — the KEM silently falls back to mere passive (CPA) security.

An **active attacker** feeds clever ciphertexts to the decapsulation oracle and tries to learn the secret: exactly the chosen-ciphertext attack the check was there to stop.

This is not hypothetical:

For 19 months, HQC's reference implementation effectively skipped the re-encryption check — and basic correctness tests still passed.

Open Quantum Safe project. Correctness error in HQC decapsulation. Reported by Célian Glénaz and Dahmun Goudarzi. CVE-2024-54137 (Dec 2024), https://github.com/open-quantum-safe/liboqs/security/advisories/GHSA-gpf4-vrrw-r8v7

Attack 2: why nobody noticed

- Two interacting bugs in HQC's clever hand-crafted constant-time code bypassed the check.
- *Honest* ciphertexts still decapsulated correctly — so functional tests passed.
- The broken code was picked up downstream (e.g. liboqs).

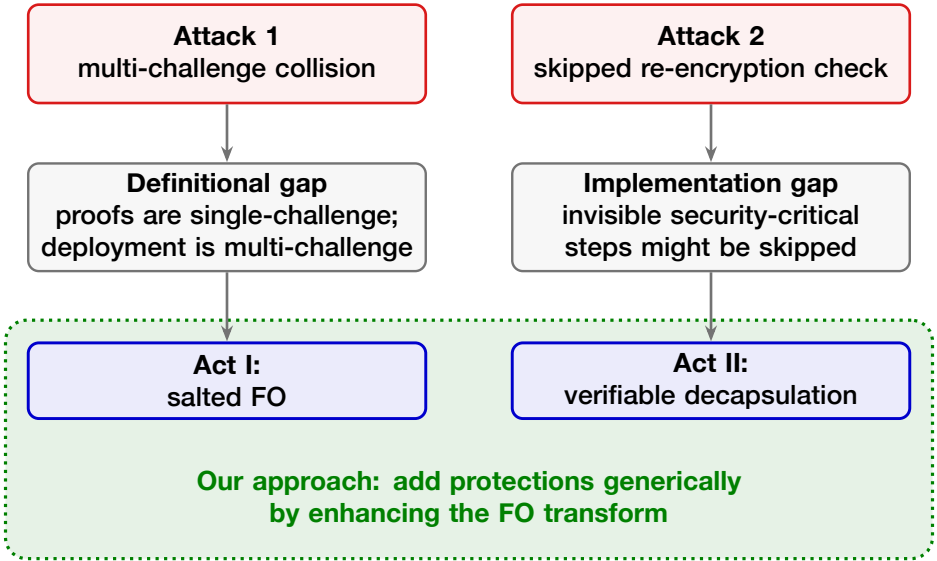
We've seen this pattern before:

Echoes Apple's "`goto fail`" (2014): one skipped verification step in TLS that was invisible to ordinary testing.

A lesson:

Danger from security-critical steps with no functional footprint.

Observations



Outline

The FO transform

Act I: Multi-target security

Act II: Verifiable decapsulation

Wrapping up

The Fujisaki–Okamoto transform in detail

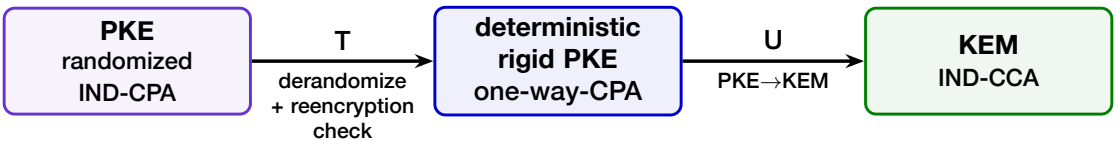
What FO promises

- Start from a PKE with only a **weak** guarantee (no protection against active attackers).
- Obtain a KEM secure against chosen-ciphertext attacks.

Fujisaki, Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *CRYPTO* 1999; *Journal of Cryptology*, 2013.

The modular view: decompose FO $=$ U $\circ$ T

Due to Hofheinz–Hövelmanns–Kiltz (2017): analyze FO as two composable steps, which perform two different roles.



Hofheinz, Hövelmanns, Kiltz. A modular analysis of the Fujisaki–Okamoto transformation. *TCC* 2017.

The T-transform: derandomization + re-encryption check

1. Derandomize:

$$\mathrm{Enc}_1(pk, m) \;:=\; \mathrm{Enc}(pk, m;\; G(m))$$

where G is a random hash function.

Each message now has *one* canonical ciphertext.

2. Re-encryption check:

1. **Decrypt:** $m' \leftarrow \mathrm{Dec}(sk, c)$
2. **Re-encrypt:** $c' \leftarrow \mathrm{Enc}_1(pk, m')$
3. **Check:** $c = c'$

The scheme becomes rigid: a ciphertext is accepted only if it is the genuine encryption of what it decrypts to.

The T-transform: rigidity

Rigidity

A ciphertext is accepted only if it is the genuine encryption of what it decrypts to.

Why does rigidity matter?

- It makes the decapsulation oracle *useless* to an attacker — it only ever decrypts honestly-formed ciphertexts.
- This is what turns *passive* (CPA) into *active* (CCA) security.

Closely connected to our motivating attacks

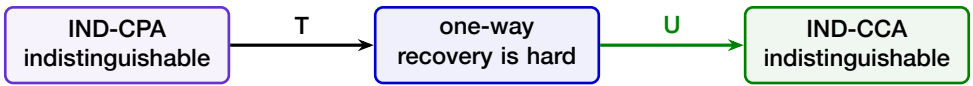
- Skipping rigidity check enabled the HQC attack.
- Determinism enabled FrodoKEM multi-ciphertext attack.

One-wayness vs. indistinguishability

Determinism costs us indistinguishability

A *deterministic* encryption can never satisfy indistinguishability (IND-CPA): to test whether c encrypts a candidate m, just recompute $\mathrm{Enc}_1(pk, m)$ and compare.

So the strongest goal we can ask of T's output is **one-wayness**:
given a ciphertext for a random message, it is hard to recover the message.



The U transform will lift us back up to indistinguishability.

The U-transform: one-way PKE $\rightarrow$ indistinguishable KEM

Encapsulation:

1. Pick random m
2. Encrypt: $c \leftarrow \mathrm{Enc}_1(pk, m)$
3. Compute shared secret: $K \leftarrow H(m, c)$

Decapsulation of c:

1. Decrypt: $m' \leftarrow \mathrm{Dec}_1(sk, c)$ (with re-encryption check)
2. Compute shared secret: $K \leftarrow H(m', c)$

Why does hashing give us indistinguishability?

- Because the message is hard to *recover* (one-wayness), the attacker can never *query* the random oracle on it.
- So output $H(m, c)$ is a fresh, uniformly random value to the adversary.

The U-transform: one-way PKE $\rightarrow$ indistinguishable KEM

What to do with an invalid ciphertext during decapsulation?

$U^{\perp}$: explicit rejection

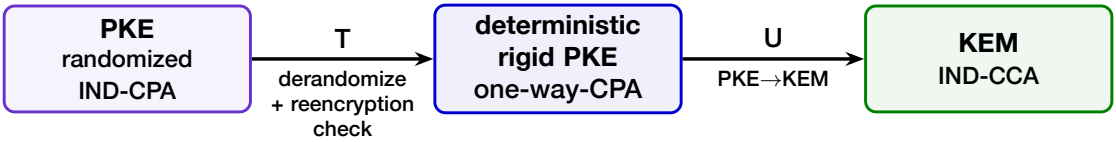
1. **Decrypt:** $m' \leftarrow \mathrm{Dec}_1(sk, c)$ **(with re-encryption check)**
2. **If $m' = \perp$, return $\perp$**
3. **Else: Compute shared secret:** $K \leftarrow H(m', c)$

$U^{\not\perp}$: implicit rejection

1. **Decrypt:** $m' \leftarrow \mathrm{Dec}_1(sk, c)$ **(with re-encryption check)**
2. **If $m' = \perp$, return $H'(z, c)$ for hidden secret z**
3. **Else: Compute shared secret:** $K \leftarrow H(m', c)$

Every standardized PQ KEM uses implicit rejection.

Summary of the FO transform



Theorem (FO security informally)

If the PKE is IND-CPA-secure, then the KEM FO $=$ U $\circ$ T is IND-CCA-secure, in the (quantum) random oracle model.

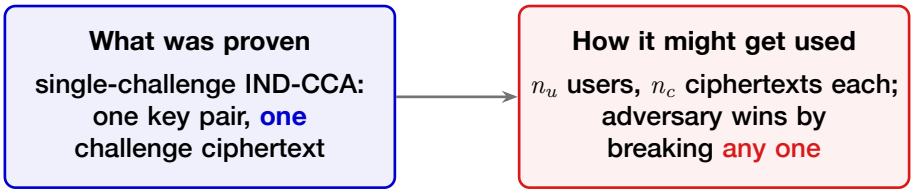
- Standardized PQ KEMs ML-KEM, FrodoKEM, HQC each use a different FO variant, but all follow the overall T-then-U pattern.

Attack #1: FO yields single-challenge security, not multi-challenge security.
Attack #2: FO only provides security if re-encryption check done properly.

Act I: Multi-target security and the salted FO transform

Lewis Glabush, Kathrin Hövelmanns, Douglas Stebila. On the multi-target security of post-quantum KEMs. *IACR Communications in Cryptology*, 3(1), May 2026.

The definitional gap



One can give a generic hybrid argument from single-challenge to multi-challenge, but that loses a factor of $n_u n_c$ which appreciably decreases security.

Can we do better?

The multi-ciphertext attack, reviewed

Adversary is given:

Public key pk

n_c **challenge ciphertexts** $c_1, \ldots, c_{n_c}$ **encrypting random messages** m_i**:**

$$c_i \leftarrow \mathrm{Enc}_1(pk, m_i) = \mathrm{Enc}(pk, m_i; G(m_i))$$

The attack:

1. **Sample** N **messages** $m'_1, \ldots, m'_N$**;**
2. **Encrypt each (***this is possible!* **after T, encryption is deterministic and** pk **is public)**

$$c'_j \leftarrow \mathrm{Enc}_1(pk, m'_j) = \mathrm{Enc}(pk, m'_j; G(m'_j))$$

3. **Store** (c'_j, m'_j)
4. **Look for a match with a challenge: if** $c'_j = c_i$**, then** $m_i = m'_j$

The multi-ciphertext attack, reviewed

$$\mathrm{Prob}[\text{successful attack}] \approx \frac{N n_c}{|\mathcal{M}|}.$$

For $|\mathcal{M}| = 2^{128}$, given $n_c = 2^{64}$ challenge ciphertexts, an adversary who does $N = 2^{64}$ work succeeds with high probability.

(This doesn't affect single challenge security $n_c = 1$.)

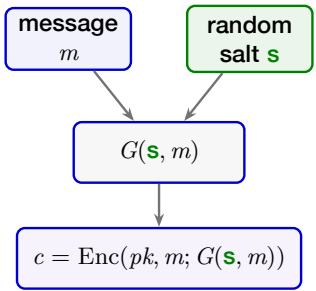
How to fix it?

Idea 1: increase $|\mathcal{M}|$

- ☹ For FrodoKEM, increasing message size leads to **much** bigger parameters

Idea 2: increase unpredictability with a random public salt

The fix: add a public salt in the T transform

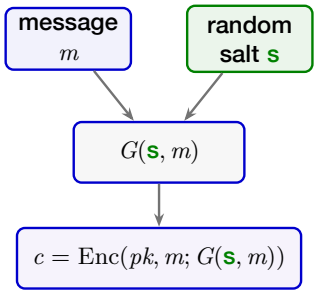


transmit $(c, \mathbf{s})$
shared secret $K = H(m, c, \mathbf{s})$

Salted FO (SFO):

1. Pick random m
2. Sample a fresh random salt s
3. Encrypt: $c \leftarrow \mathrm{Enc}(pk, m; G(\mathbf{s}, m))$
4. Transmit $(c, \mathbf{s})$
5. Compute shared secret: $K = H(m, c, \mathbf{s})$.

The fix: add a public salt in the T transform



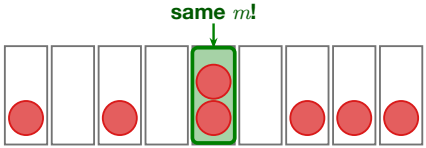
transmit $(c, \mathbf{s})$
shared secret $K = H(m, c, \mathbf{s})$

For a collision to help, the adversary must now also match the salt — which is information-theoretically random. Hard to guess in advance; defeats batch and precomputation attacks.

$$\mathrm{Prob}[\textbf{successful attack}] \approx \frac{N\, n_c}{|\mathcal{M}|\, 2^{\mathrm{len_s}}}.$$

Bottom line: if $\mathrm{len_s} \geq 256$ (32 bytes of extra communication), we thwart this multi-ciphertext attack.

Wait, there's more: an almost-free collision



• challenge ciphertexts $c_1, \ldots, c_{n_c}$ for messages selected randomly (with replacement)

$$\text{Prob}[\textbf{collision}] \approx \frac{n_c^2}{|\mathcal{M}|}$$

If two challenges encrypt the same message, they lead to the same key and ciphertext:

$$m_i = m_j \;\Rightarrow\; c_i = c_j, \quad K_i = K_j$$

But *independent* random keys would not match.

$\Longrightarrow$ Adversary can distinguish real vs. random challenge set without any queries!

Salts fix this too!

Salts reduce these types of collisions:

$$\text{Prob}[\textbf{collision}] \approx \frac{n_c^2}{|\mathcal{M}|\, 2^{\text{len}_s}}$$

Salted FO results

Theorem: Salted FO is tightly multi-target secure in the classical random oracle model.

$$\mathbf{Adv}^{\mathsf{IND\text{-}CCA}}_{n_c,n_u}(\mathcal{A}) \;\leq\; \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{n_c,n_u}(\mathcal{B}) \;+\; \frac{n_u\, n_c^2}{|\mathcal{M}|\; 2^{\mathsf{len_s}}} \;+\; \frac{9\; q_{\mathsf{RO}}}{|\mathcal{M}|} \;+\; q_{\mathsf{RO}} \cdot \delta.$$

multi-ciphertext (n_c), multi-user* (n_u), δ-multi-user correctness

Tight reduction: multi-target IND-CPA of PKE $\implies$ multi-target IND-CCA of KEM.

The dangerous $n_u n_c$-type terms are now divided by $2^{\mathsf{len_s}}$ — the salt buys back security.

What about quantum attackers?

A quantum random oracle model (QROM) proof exists — but is *non-tight*.

* Multi-user security addressed by hashing the public key into shared secret derivation: Duman, Hövelmanns, Kiltz, Lyubashevsky, Seiler. Faster lattice-based KEMs via a generic Fujisaki–Okamoto transform using prefix hashing. *ACM CCS* 2021.

Multi-target security, salted vs. unsalted

Unsalted FO	$\dfrac{n_u n_c^2 + (2n_c + 1)\, q_{\mathsf{RO}}}{\lvert\mathcal{M}\rvert}$	≈ 1 in the FrodoKEM / HQC regime
Salted FO	$\dfrac{n_u n_c^2}{\lvert\mathcal{M}\rvert\, 2^{\mathsf{len_s}}} + \dfrac{9\, q_{\mathsf{RO}}}{\lvert\mathcal{M}\rvert}$	negligible for salt length $\approx$ 32 bytes

Bottom line

At the 128-bit level, replacing FO with Salted FO recovers ~ 62 bits of multi-target security for FrodoKEM and HQC.

Cost: one extra public salt per ciphertext; $< 1\%$ runtime / bandwidth overhead.

FrodoKEM ISO standard uses Salted FO.

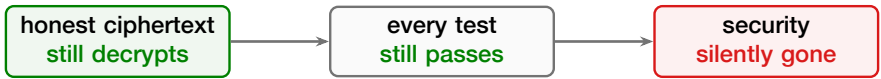
Act II: Verifiable decapsulation

Lewis Glabush, Felix Günther, Kathrin Hövelmanns, Douglas Stebila. Verifiable decapsulation: recognizing faulty implementations of post-quantum KEMs. *CRYPTO* 2025.

Lewis Glabush, Felix Günther, Britta Hale, Elizabeth Van Oorschot, Douglas Stebila. Augmenting post-quantum KEMs for verifiable decapsulation with rejection hiding. Under review, 2026.

Brittle cryptography

- HQC's re-encryption check was effectively skipped — unnoticed for 19 months.
- Why was it missed?
  - Skipping it breaks nothing visible: honest ciphertexts decrypt, all tests pass.
  - Only the security is lost.



Principle (Heninger 2019; Fischlin–Günther 2023)

Tie security to functionality: a faulty-but-benign implementation should break something *visible*.

Heninger. Biased Nonce Sense. Invited talk, *Workshop on Attacks in Cryptography (WAC2)*, 2019.
Fischlin, Günther. Verifiable verification in cryptographic protocols. *ACM CCS* 2023.

“Verifiable verification” and confirmation codes

Fischlin and Günther’s **Verifiable verification**:
Modify the scheme so its output changes if security-critical verification steps are skipped.

Algorithm produces a **confirmation code** cd:

- An extra deterministic output, unpredictable if a security-critical computation is omitted.
- Skip the step $\implies$ unlikely to produce a valid cd.
- Formal notion: *confirmation-code unpredictability* (cUP).

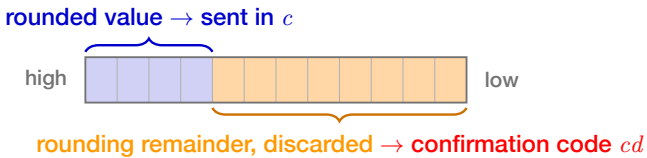
Confirmation-code unpredictability advantage doesn’t have to be cryptographically good

Probability $\frac{1}{2}$ suffices: a faulty implementation that fails half the time will still be noticed in basic interop testing.

Fischlin, Günther. Verifiable verification in cryptographic protocols. *ACM CCS* 2023.

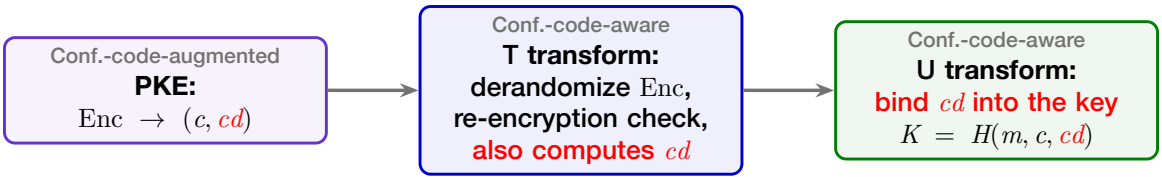
Confirmation code for ML-KEM

- ML-KEM ciphertexts are compressed versions of polynomials in $\mathbb{Z}_{3329}[X]/(X^{256}+1)$
- **Coefficient-wise compression** rounds $\mathbb{Z}_{3329}$ to $\mathbb{Z}_{2^d}$ ($d = 4, 5, 10,$ or 11)
- Not exactly bit truncation (since q odd), but intuitively similar:



- Remainder is unpredictable without re-doing most of the encryption computation.
  - Proof technique: model pseudorandom generator in random oracle model.
- Since the remainder is computed during encryption, then thrown away, keeping $12-20$ bytes for cd is effectively free.

Confirmation code-augmented FO transform



cd is never transmitted: decapsulation can only recompute it by actually re-encrypting.

Still IND-CCA-secure.

Faulty decapsulation implementation (skipping re-encryption and hence code re-computation) now yields a wrong key (faulty correctness (fCOR)).

Formalization

Confirmation-code unpredictability:

Without doing the security-critical computation, an adversary cannot produce a valid code cd.

Faulty correctness:

An implementation that skips the security-critical step produces the *wrong* shared secret with noticeable probability.

Theorem: Confirmation-code-augmented FO is secure.

Let FOC $=$ UC $\circ$ TC applied to PKE.

- If PKE is IND-CPA-secure, then FOC is IND-CCA-secure. (ROM, QROM)
- If PKE is cUP-secure, then FOC is fCOR-secure. (ROM)

Application to HQC and ML-KEM

Scheme	Confirmation code cd	Computational overhead	Guessing probability
HQC	1 byte from end of pre-truncation ciphertext	0.25%	0.03
ML-KEM	$12 - 20$ bytes of discarded compression bits	$1.2 - 3.4\%$	$0.17 - 0.68$

Catching the original bug

With a confirmation code, HQC's reference-implementation bug surfaces on the first interop test, not 19 months later.

We've proposed to NIST that this be included in the final HQC spec.
(Too late for ML-KEM.)

One drawback

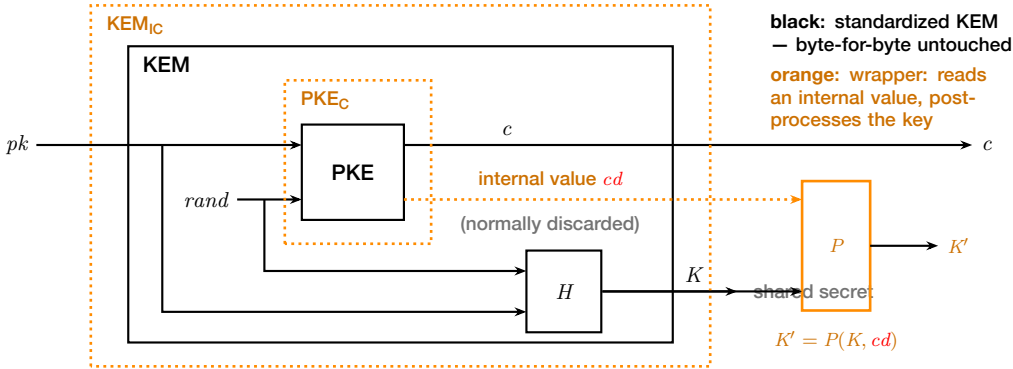
Hashing cd into the key *changes* the scheme

- Shared secret is different from standard since it now depends on cd
- Hard sell after standardization.

Our solution:

- KEM computations left untouched – as standardized.
- Wrapper reads *internal* values to pull out confirmation code.
- Post-process original shared secret + confirmation code.

Wrapping the standardized KEM



Variants: implicit versus explicit rejection, whether confirmation code is considered internal or public.

This figure shows encapsulation for **internal-code** variant.

Rejection hiding

We had to make sure the confirmation codes didn't undermine the implicit or explicit rejection goals of $U^{\not\perp}$ and $U^{\perp}$.

But... what does implicit rejection actually mean?

All major PQC KEMs (ML-KEM, HQC, FrodoKEM, Classic McEliece) use implicit rejection, but it was never formally defined.

We define **rejection hiding** and prove that Hofheinz–Hövelmanns–Kiltz's original $FO^{\not\perp}$ transform satisfies it, plus our confirmation-code-augmented variants.

Wrapping up

Two themes

The multi-target gap

Standard definitions don't necessarily match deployment reality.

Approach: a structural tweak (a **salt**) with a *tight* reduction.

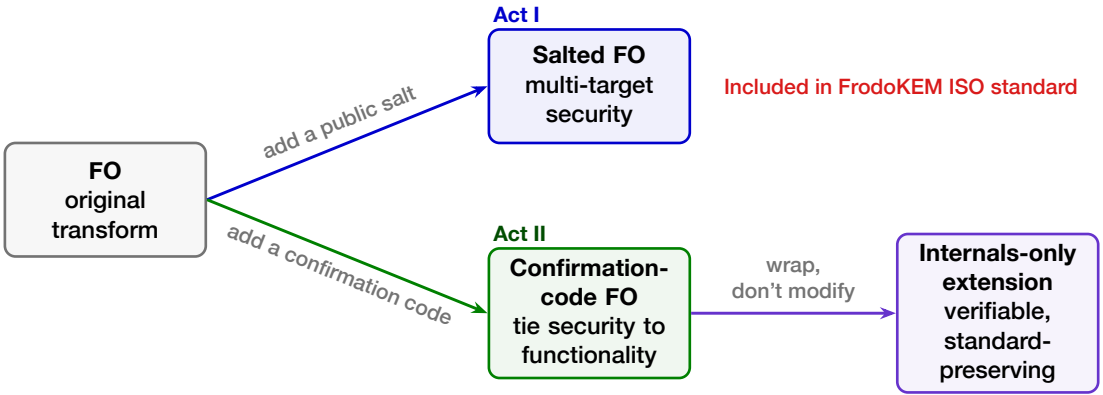
The implementation gap

Implementations can have flaws that may affect security.

Approach: make security-critical steps **functionally observable**.

Both are about closing the distance
between what is proven and how it gets used.

Additional functionality via the Fujisaki–Okamoto transform



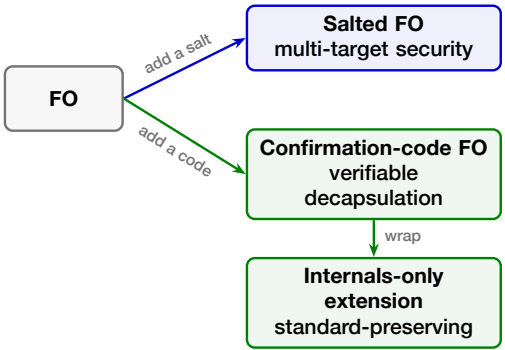
Open questions

- **Tight QROM bounds** for salted FO.
- **Lower bounds** on salt length and confirmation-code length.
- A **combined** multi-target + verifiable variant.
- Verifiable decapsulation for **explicitly-rejecting** deterministic PKEs — no proofs known.
- Re-encryption can create side-channel vulnerabilities – can we get rid of re-encryption altogether? Seems hard.[†]

[†] Hövelmanns, Hülsing, Majenz, Sisinni. (Un)breakable curses - re-encryption in the Fujisaki-Okamoto transform. *Eurocrypt* 2025.

Adding functionality to post-quantum cryptography with variants of the Fujisaki–Okamoto transform

Douglas Stebila, University of Waterloo



- A public **salt** restores multi-target security (now in FrodoKEM ISO standard)
- **Confirmation codes** expose faulty implementations (proposed for HQC)
- Apply generically via variants of FO transform
- Closing the gap between what is proven and how it gets used

Glabush, Hövelmanns, Stebila. On the multi-target security of post-quantum KEMs. *IACR Communications in Cryptology*, 3(1), May 2026.
Glabush, Günther, Hövelmanns, Stebila. Verifiable decapsulation: recognizing faulty implementations of post-quantum KEMs. *CRYPTO* 2025.
Glabush, Günther, Hale, Van Oorschot, Stebila. Augmenting post-quantum KEMs for verifiable decapsulation with rejection hiding. Under review, 2026.

Appendix

Anatomy of the HQC bug

Bug 1: re-encryption reads the public key from the wrong offset in sk

```
- const uint8_t *pk = sk + SEED_BYTES;
+ const uint8_t *pk = sk + SEED_BYTES + VEC_K_SIZE_BYTES;
```

Re-encrypts under garbage $\Rightarrow$ the comparison with (u, v) *always fails*.

Bug 2: the constant-time selection mask is inverted

```
- result = (uint8_t) (-((int16_t) result) >> 15);
+ result -= 1;
  mc[i] = (m[i] & result) ^ (sigma[i] & ~result);
```

Old mask selects m (= accept) exactly when the comparison *fails*.

Two wrongs make a silent right

Always-failing check $\times$ inverted acceptance $=$ always accept. Either bug alone fails correctness tests instantly; together they pass every functional test — only the security is gone.

liboqs commit `cce1bfde`; CVE-2024-54137; reported by Célian Glénaz and Dahmun Goudarzi.
https://github.com/open-quantum-safe/liboqs/commit/cce1bfde4e52c524b087b9687020d283fbde0f24

Implicit vs. explicit rejection — security history

- **Implicit** rejection: proofs available first (HHK 2017), and tighter.
- **Explicit** rejection: early treatments added a key-confirmation hash; first QROM proofs (Don–Fehr–Majenz–Schaffner 2021) had bigger tightness loss.
- Hövelmanns–Hülsing–Majenz 2022 (“Failing gracefully”): bounds closer to implicit, for *probabilistic* PKEs with certain correctness properties.
- Hövelmanns–Kudinov (eprint 2025/062): explicit rejection essentially as secure as implicit, in the extractable QROM.
- Remaining gap: *deterministic* PKE + explicit rejection — no proofs known (the open question from the closing slide).

Multi-target IND-CCA definition

$\underline{\mathrm{Exp}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}}_{\mathsf{KEM}}(\mathcal{A})}$

01 $b \leftarrow\$ \{0,1\}$
02 **for** $j = 1, ..., n_u$
03 $\quad (pk_j, sk_j) \leftarrow\$ \mathsf{Gen}()$
04 $\vec{pk} = (pk_1, ..., pk_{n_u})$
05 $b' \leftarrow \mathcal{A}^{\textsc{Decaps},\textsc{Chall}_1,...,\textsc{Chall}_{n_u}}(\vec{pk})$
06 **return** $[\![b = b']\!]$

$\underline{\textsc{Chall}_j}$ // At most n_c queries
07 $(c, \mathbf{ss}_0) \leftarrow\$ \mathsf{Encaps}(pk_j)$
08 $\mathbf{ss}_1 \leftarrow\$ \mathcal{K}$
09 $\mathfrak{L}_{C_j} \leftarrow \mathfrak{L}_{C_j} \cup \{c\}$
10 **return** $(c, \mathbf{ss}_b)$

$\underline{\textsc{Decaps}(j, c \notin \mathfrak{L}_{C_j})}$
11 **return** $\mathsf{Decaps}(sk_j, c)$

- n_u independent key pairs; n_c challenge ciphertexts per user.
- Adversary gets decapsulation oracles for all users (except on challenge ciphertexts).
- Wins by distinguishing *any* challenge key from random.

The salted FO bound, general form

$$\mathbf{Adv}^{\mathbf{IND\text{-}CCA}}_{n_c,n_u}(\mathcal{A}) \;\le\; \mathbf{Adv}^{\mathbf{IND\text{-}CPA}}_{n_c,n_u}(\mathcal{B}) \;+\; \frac{n_u\, n_c(n_c-1)}{|\mathcal{M}|\; 2^{\mathbf{len_s}}} \;+\; \frac{(2t+1)\; q_{\mathbf{RO}}}{|\mathcal{M}|} \;+\; q_{\mathbf{RO}} \cdot \delta(n_u)$$

- $q_{\mathbf{RO}}$: total number of random-oracle (hash) queries by the adversary.
- $\delta(n_u)$: multi-user correctness error of the PKE.
- t: small constant from the proof; $t \le 4$ for FrodoKEM-640 parameters, giving the "9" on the main slide.

The QROM bound for salted FO

In the quantum random oracle model, the adversary may query the hash function *in superposition*. Current proof techniques lose square-root factors.

$$\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}_{\mathsf{KEM}}(\mathcal{B}) \le 4\sqrt{d \cdot \mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{A})} + \frac{n_u n_c (n_c - 1)}{|\mathcal{M}| \cdot 2^{\mathsf{len}_{\mathsf{salt}}}} + 4\sqrt{\frac{dt(q_H + q_G)}{|\mathcal{M}|}} \ ,$$

Whether the quadratic loss reflects a real quantum attack or is an artifact of the proof technique is open — for salted FO and for plain FO alike.

d: Query depth

cUP and fCOR definitions

Confirmation-code unpredictability (cUP)

$\underline{\mathrm{Exp}^{\mathsf{cUP}}_{\mathsf{PKE_C},\mathsf{F},\mathsf{max}}(\mathcal{A})}$

01 $(\mathsf{pk},\mathsf{sk}) \leftarrow\!\$\ \mathsf{KeyGen}^{\mathsf{F}}()$
02 $(m, r) \leftarrow\!\$\ \mathcal{M} \times \mathcal{R}$
03 $(c, \mathsf{cd}) \leftarrow \mathsf{Enc}^{\mathsf{F}}_{\mathsf{C}}(\mathsf{pk}, m; r)$
04 $\mathsf{cd}' \leftarrow \mathcal{A}^{\bar{\mathsf{F}}}(\mathsf{pk}, \mathsf{sk}, c, m, r)$
05 **return** $[\![\mathsf{cd} = \mathsf{cd}']\!]$

$\underline{\bar{\mathsf{F}}(x)\ /\!/\text{at most } \mathsf{max} \text{ many queries}}$

06 **return** $\mathsf{F}(x)$

Faulty implementation correctness (fCOR)

$\underline{\mathrm{Exp}^{\mathsf{fCOR}}_{\mathsf{KEM},\mathsf{F},\mathsf{max}}(\mathcal{A})}$

01 $(\mathsf{pk},\mathsf{sk}) \leftarrow\!\$\ \mathsf{KEM.KeyGen}^{\mathsf{O},\mathsf{F}}()$
02 $(c, K_0) \leftarrow\!\$\ \mathsf{KEM.Encaps}^{\mathsf{O},\mathsf{F}}(\mathsf{pk})$
03 $K_0' \leftarrow \mathsf{KEM.Decaps}^{\mathsf{O},\mathsf{F}}(\mathsf{sk}, c)$
04 **if** $K_0' \neq K_0$ // correctness error
05 **return** 1 // is a trivial win
06 $K_1 \leftarrow \mathcal{A}^{\mathsf{O},\bar{\mathsf{F}}}(\mathsf{pk}, \mathsf{sk}, c)$
07 **return** $[\![K_0 = K_1]\!]$

$\underline{\bar{\mathsf{F}}(x)\ /\!/\text{at most } \mathsf{max} \text{ many queries}}$

08 **return** $\mathsf{F}(x)$

Faulty implementation correctness (fCOR)

Definitions from Glabush, Günther, Hövelmanns, Stebila. Verifiable decapsulation. *CRYPTO* 2025.

Confirmation codes for ML-KEM and HQC: the numbers

ML-KEM: confirmation-code guessing probabilities

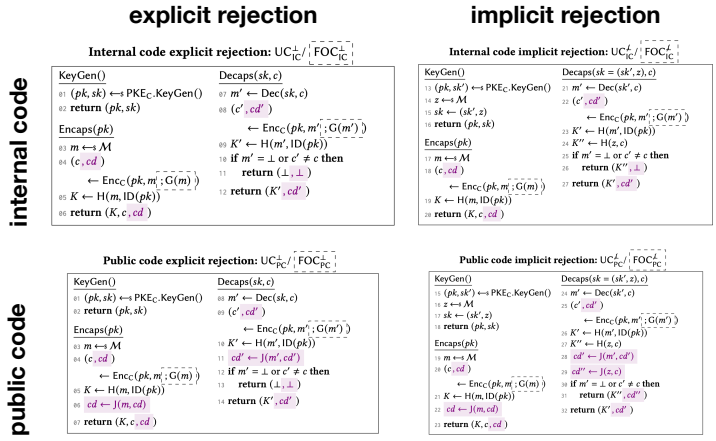
Scheme	Conditional entropy of $\mathbf{e}_1$ coeff. (h_{e_1})	Conditional entropy of e_2 coeff. (h_{e_2})	Conf. code guessing probability Short ($\lvert S\rvert = 2$) (p_2)	Conf. code guessing probability Full ($\lvert S\rvert = 256$) (p_{256})
Assuming uniform z				
ML-KEM-512	1.2733	2.0188	0.1712	2^{-325}
ML-KEM-768	1.2733	2.0188	0.1712	2^{-325}
ML-KEM-1024	0.6407	2.0069	0.4114	2^{-164}
Worst-case analysis				
ML-KEM-512	0.3219	0.3219	0.6806	2^{-71}
ML-KEM-768	0.3219	0.3219	0.6806	2^{-71}

ML-KEM & HQC: performance overhead

Algorithm		Short conf. code Size (bytes)	Short conf. code Slowdown	Full-length conf. code Size (bytes)	Full-length conf. code Slowdown
ML-KEM-512	Encaps	12	3.4%	1152	23.6%
	Decaps		3.1%		23.6%
ML-KEM-768	Encaps	16	2.1%	1536	20.5%
	Decaps		2.1%		15.0%
ML-KEM-1024	Encaps	20	1.2%	1920	14.1%
	Decaps		1.6%		12.2%
HQC-128 with bug	Encaps	1	0.09%	—	—
	Decaps		0.25%		
HQC-128 bug fixed	Encaps	1	0.13%	—	—
	Decaps		0.19%		

Tables from Glabush, Günther, Hövelmanns, Stebila. Verifiable decapsulation. *CRYPTO* 2025.

Internals-only conf.-code-augmented FO transform variants



Public-code variants mask the code with an extra hash so that it does not leak the rejection event.

Figures from Glabush, Günther, Hale, Van Oorschot, Stebila. Augmenting post-quantum KEMs for verifiable decapsulation with rejection hiding. Under review, 2026.

Rejection hiding definition

$\underline{\mathrm{Exp}^{\text{RH-ATK}}_{\mathrm{KEM}}(\mathcal{A})}$

$(pk, sk) \leftarrow\!\$\ \mathrm{KeyGen}()$
$b \leftarrow\!\$\ \{0,1\}$
$(K_0, c_0) \leftarrow\!\$\ \mathrm{Encaps}^{\mathrm{G}}(pk)$
$b' \leftarrow\!\$\ \mathcal{A}^{\mathrm{Chall}, O_{\mathrm{G}}, O_{\mathrm{Decaps}}}(pk, c_0)$
if $c_1 \in \mathcal{L}_G \cup \mathcal{L}_D$:
  $b' \leftarrow 0$ // adversary loses
return $[\![b' = b]\!]$

$\underline{\mathrm{Chall}(c \neq c_0)\ //\ \text{queried exactly once}}$

$c_1 \leftarrow c$
$K' \leftarrow \mathrm{Decaps}^{\mathrm{G}}(sk, c_b)$
return K'

$\underline{O_{\mathrm{G}}(m)}$

$r \leftarrow \mathrm{G}(m)$
$(c, K) \leftarrow \mathrm{Encaps}^{\mathrm{G}}(pk; m)$
$\mathcal{L}_G \leftarrow \mathcal{L}_G \cup \{c\}$
return r

$\underline{O_{\mathrm{Decaps}}(c \neq c_0)}$

$K' \leftarrow \mathrm{Decaps}^{\mathrm{G}}(sk, c)$
$\mathcal{L}_D \leftarrow \mathcal{L}_D \cup \{c\}$
return K'

- Adversary must distinguish whether decapsulation of an invalid ciphertext *rejected* — given the full output (shared secret, and code if public).
- Theorem: HHK's original FO$^{\not\perp}$ is RH-CCA in the ROM, with

$$\mathbf{Adv}^{\text{RH-CCA}}_{\mathbf{KEM}} \leq \mathbf{Adv}^{\text{IND-CCA}}_{\mathbf{KEM}} + \frac{q_H}{|\mathcal{M}|} + 2^{-\gamma} + \delta$$