

On the Instantiability of the Fujisaki–Okamoto Transform for Trapdoor Functions

Carter Luck
University of Massachusetts Amherst
cluck@umass.edu

Xinyu Mao
University of Waterloo
xinyumao.tcs@gmail.com

Adam O’Neill
University of Massachusetts Amherst
adamoneill@umass.edu

Abstract

The Fujisaki–Okamoto (FO) transform (Journal of Cryptology, 2013) is the standard route from weakly secure public-key encryption to chosen-ciphertext security, and it underlies many practical post-quantum key-encapsulation schemes, including ML-KEM. Security proofs for FO are set in the (quantum) random oracle models. FO uses two hash functions: the derandomization hash H , which turns the base scheme into a deterministic trapdoor function (TDF) by Encrypt-with-Hash, and the key-derivation hash G . We ask whether it is possible to instantiate the hash functions in the FO transform while retaining its security guarantees.

While prior work mainly focused on the H step, our focus is the G step, and we take an underlying injective TDF as given. We first show that the message-only implicit-rejection variant U_m^L , a key-encapsulation mechanism formulation of FO that follows ML-KEM’s choices of rejection and key derivation, is *uninstantiable* in general. Namely, assuming standard LWE and subexponentially secure indistinguishability obfuscation and puncturable PRFs, for every efficient hash function family G with superlogarithmic output length there is a one-way, perfectly correct injective trapdoor function for which the resulting KEM is not IND-CCA2 secure.

We next give sufficient conditions on G to instantiate the TDF-based *hybrid-encryption* formulation of FO, rather than the KEM version. To this end, we introduce *distributional injective extractability* (DINJEXT), a new hash function property defined relative to a partially injective “wrapping function.” Under this assumption relative to the FO-derived wrapping function, combined with an achievable form of universal computational extractor security (Bellare, Hoang, and Keelveedhi, CRYPTO 2013) on G and suitable assumptions on the base schemes, we obtain standard-model IND-CCA2 security. As evidence that DINJEXT is achievable, we show that a random oracle satisfies it for every partially injective wrapping function, and that a quantum random oracle satisfies a formulation of it for the FO wrapping function, in both cases even relative to an oblivious sampling oracle. We leave a standard-model construction open.

Contents

1	Introduction	3
1.1	Background and Motivation	3
1.2	Main Results	3
1.3	Uninstantiability of the G Step for FO as a Key Encapsulation	4
1.4	Instantiability of the G Step for FO as Hybrid Encryption	5
1.5	Related Work	8

2	Preliminaries	9
2.1	Notation	9
2.2	Encryption Schemes and Key Encapsulation	9
2.3	Trapdoor Functions	11
2.4	Two Properties of Symmetric Encryption	13
2.5	Hash Functions and Universal Computational Extractors	14
2.6	Obfuscation and Leftover Hash Lemma	16
3	The Fujisaki–Okamoto Transform	19
4	A Negative Result for $U_m^\perp$	20
4.1	Building the Counterexample	20
4.2	The Attack	21
4.3	One-Wayness	24
4.4	Supporting Proofs	25
5	Injective Extractability	33
5.1	Coin-Based Injective Extractability	34
5.2	Impossibility of Post-Quantum Coin-Based Extractability	35
5.3	Distributional Injective Extractability	37
5.4	The (Distributional) Injective Extractability Hierarchies	38
5.5	Negative Results and an Open Problem	40
6	A Positive Result for FO Hybrid Encryption	41
6.1	The FO Wrapping Function	41
6.2	The Fujisaki–Okamoto CPA-to-CCA Lifting Theorem	42
6.3	IND-CPA Security of FO Hybrid Encryption from UCE	49
7	Distributional Injective Extractability in the ROM and QROM	51
7.1	Modeling in the ROM and QROM	51
7.2	Achievability Results in the ROM and QROM	55
A	Modular Composition for DINJEXT and UCE	62
B	Negative Results for 1-DINJEXT2	66
B.1	Input Channel Counterexample	67
B.2	Witness Channel Counterexample	68
B.3	Setup Channel Counterexample	69
B.4	Challenge Image Counterexample	70
C	Proofs for the ROM and QROM Results	76
C.1	Proof of Proposition 7.3	76
C.2	Proof of Theorem 7.4	78

1 Introduction

1.1 Background and Motivation

The Fujisaki–Okamoto transform. The Fujisaki–Okamoto (FO) transform [46] is a central method for obtaining public-key encryption secure against chosen-ciphertext attack. It underlies Kyber and the derived ML-KEM standard [21, 66]. In the random oracle model (ROM) [14], FO converts suitably secure public-key encryption into encryption secure against chosen-ciphertext attack; variants also admit proofs in the quantum random oracle model (QROM). We study which standard-model properties of a hash function can replace the features of the random oracle used in these proofs.

Random-oracle instantiability. The ROM models a cryptographic hash function as a random function accessible to all parties via an oracle. The study of instantiability goes back to Canetti, Goldreich, and Halevi [32] and to many follow-on works showing that there are (contrived) ROM schemes that become insecure when the random oracle is replaced by any concrete function. Recent work has given more natural examples [44, 61]. A parallel line of work initiated by Canetti [27] studies what standard-model properties a hash function must satisfy to replace a random oracle in specific cases. That is, for each ROM proof one needs to study which property its use of the random oracle actually needs and whether that property is achievable. Our negative result belongs to the first line of research, and our positive result belongs to the second.

Two uses of hashing in FO. In the case of FO, hashing is used in two different ways. Starting from randomized public-key encryption, a hash function H derives the encryption coins from a sampled plaintext x . This is the Encrypt-with-Hash [7] step, which defines the deterministic map

$$x \mapsto \text{PKE.Enc}(pk, x; H(K_H, x)).$$

A second hash function G derives a session key from x and, in some variants, the resulting public-key ciphertext. We call these two steps the H step and the G step. If one starts directly from a suitable trapdoor function, such as in the case of Classic McEliece [4] or NTRU [35], the H step is skipped entirely.

Whereas prior uninstantiability results concern the H step [25, 49, 75], our focus is the G step. We take the underlying injective trapdoor function (TDF) as given, including one obtained through the H step, as in Kyber [21], provided it satisfies the hypotheses of our result. The prior H -step impossibility results do not establish insecurity for a fixed scheme such as Kyber. It is therefore meaningful to assume the required properties of the resulting TDF and ask what is needed of G . Our motivating question is:

What properties of the key-derivation hash function G suffice to instantiate the G step of FO in the standard model?

1.2 Main Results

Overview. We give complementary negative and positive results. We first show that the G step is *uninstantiable* in general: under the assumptions stated below, no fixed hash function family G with superlogarithmic output length makes the implicit-rejection key-encapsulation mechanism

(KEM) variant of FO secure for every one-way injective TDF. Here, the session key is derived from the TDF preimage alone, and the result holds even when the TDF is perfectly correct. Thus, no property of G on its own gives a standard-model security theorem for this KEM formulation of FO that covers every one-way injective TDF.

For our positive results, we adopt the view of FO as a hybrid encryption scheme, in which the derived key encrypts the message under an authenticated symmetric scheme. Murphy, O’Neill, and Zaheri [65] showed this setting to be beneficial for standard-model instantiation versus the KEM setting. Our negative result does not apply to it, because the attack tests a candidate encapsulated key and an adversary against hybrid encryption never receives one. In this setting we prove a modular, standard-model CPA-to-CCA lifting theorem. Namely, under standard assumptions on the base schemes, IND-CCA2 security of FO follows from two properties of G : an achievable form of universal computational extractor (UCE) security [10, 36], and a new property we call *distributional injective extractability* (DINJEXT).

More on distributional injective extractability. DINJEXT is not a property of G alone. We define it relative to a partially injective *wrapping function*, which combines a preimage x , the hash of x , and auxiliary data into an image. The FO wrapping function produces the ciphertext from the TDF preimage x and the message, using the hash $G(K_G, x)$ as the symmetric key, so the assumption is joint on the base primitives as well as on G . At a high level, DINJEXT translates the *observability* property of a random oracle, in which the simulator watches the adversary’s query log, to the standard model. It is distributional in that it asks for simulation of the adversary’s interaction with an ideal inversion oracle rather than extraction from the adversary’s coins. This is what makes it post-quantum compatible, which is what the FO setting needs, as well as weaker than conventional coin-based extractability even in the classical setting. We describe the notion and the FO wrapping function in Section 1.4.

While we apply DINJEXT only to FO, we expect it and the distributional approach behind it to have other applications. Candidates include instantiating, in the standard model, other ROM transforms whose random-oracle proofs rely on observability, such as OAEP and OAEP+, and addressing post-quantum knowledge-soundness for proofs of knowledge.

1.3 Uninstantiability of the G Step for FO as a Key Encapsulation

We show that no standard-model hash function G with superlogarithmic output length makes the KEM $U_m^\perp[F, G]$ secure for every injective one-way TDF F , assuming standard LWE together with subexponentially secure indistinguishability obfuscation (iO) and puncturable PRFs. Namely, for each concrete hash function G , we construct an injective TDF $F_G^\star$ for which $U_m^\perp[F_G^\star, G]$ is not CCA-secure (Theorem 4.1), where $U_m^\perp$ denotes a variant of the FO transform described below.

Which variant of FO we attack. The variant of FO we attack follows two choices made in ML-KEM [66].

- Implicit rejection. On an invalid input c that fails the re-encryption check, decapsulation returns a fallback key $G(K_G, Z||c)$, where Z is a secret prefix chosen during key generation.
- Message-only key derivation. The session key is $G(K_G, u)$, computed from the preimage alone, rather than $G(K_G, u||c)$, which also takes the ciphertext.

Hofheinz, Hövelmanns, and Kiltz write the resulting KEM as $U_m^L[F, G]$ for TDF F and hash function G ; we keep their notation [54]. Using implicit rejection only makes the negative result stronger, as it shows that hiding the validity bit still does not make one-wayness sufficient for instantiation.

The counterexample is tailored to the code of G ; it does not give an attack on a deployed scheme. Its significance is that no standard-model property of G alone can give a theorem for this KEM covering every one-way injective TDF. In particular, an instantiation of this variant must rely on properties of the underlying TDF beyond one-wayness, regardless of what is assumed about G .

The attack. We present a simplified version here to convey the main ideas. Let $f(\cdot, \cdot)$ be a lossy trapdoor function. The counterexample uses two key pairs (ek_0, td_0) , (ek_1, td_1) of f and public vectors a_0, a_1 chosen uniformly at random. Define

$$F_G^\star(j, x) = (j, f(ek_j, x), \langle a_j, x \rangle).$$

Inversion uses td_j and checks the recovered preimage by re-evaluation. We call the first component j the tag.

Define the predicate $P(c, K_G, K)$ to equal one if and only if c is a valid tag-zero ciphertext whose preimage $(0, x)$ satisfies $G(K_G, (0, x)) = K$. Access to P would let an adversary test the challenge key and hence break the CCA-security of $U_m^L[F_G^\star, G]$. However, directly obfuscating P makes one-wayness of $F_G^\star$ difficult to establish, since P has td_0 hardcoded. Instead, the public key contains an obfuscation of a leaky circuit C_{leak} . On input (c, K_G, K) , this (probabilistic) circuit computes $\beta = P(c, K_G, K)$ and encodes it in the pair

$$Q = (1, f(ek_1, X), \beta \oplus \langle a_1, X \rangle), \quad R = G(K_G, (1, X)),$$

where X is sampled uniformly. With access to the decapsulation oracle of $U_m^L[F_G^\star, G]$, the adversary queries it on Q and compares the answer with R . When $\beta = 0$, Q is valid and decapsulation returns R , whereas when $\beta = 1$, Q is invalid and decapsulation returns the fallback key, which for an arbitrary G can also equal R . The full attack (Section 4.2) therefore also uses a second test, applied when the challenge ciphertext has tag one, whose advantage is large when this equality is frequent.

Keeping the trapdoor function one-way. It remains to show that $F_G^\star$ is one-way, even though its public key contains an obfuscation of C_{leak} . Using the security property of probabilistic iO [33], it suffices to prove that C_{leak} hides the bit β statistically. To this end, we first switch ek_1 to lossy mode. Switching to lossy mode leaves enough uncertainty about X for the random inner product to hide β statistically, even given R . The full construction in Section 4.1 amplifies this statistical hiding property to meet the condition required by the probabilistic iO of [33], which can be based on subexponentially secure deterministic iO and subexponentially secure puncturable PRFs. The lossy TDF can be based on LWE.

1.4 Instantiability of the G Step for FO as Hybrid Encryption

The role of extractability. We first discuss the extractability property of G used in our instantiability result. Consider a decryption query (c_1, c_2) made in the IND-CCA2 game. Recovering x with $F.\text{Eval}(pk, x) = c_1$ lets the reduction derive the key $G(K_G, x)$ and decrypt c_2 . When G is a random

oracle, its query log contains the candidate preimages. However, a concrete hash function does not give any query log. We therefore seek a standard-model property that supports the simulation of decryption queries.

In the context of FO, G may be compressing, and the adversary submits a ciphertext to its decryption oracle that does not contain any hash value. Either of these makes the standard notions of extractability of Canetti and Dakdouk [31] insufficient. This is why we define extraction relative to a *wrapping function*, which for FO is the map

$$g_{pk}(x, y, m; \rho) = (\text{F.Eval}(pk, x), \text{SE.Enc}(y, m; \rho))$$

where $y = G(K_G, x)$ is the hash value, m is the message, and ρ are the encryption coins. We call the ciphertext the *image* and x its *preimage*, and extraction means recovering the preimage of a valid image. This approach builds on the QROM extractability techniques of Don, Fehr, Majenz, and Schaffner (DFMS) [41], capturing them as a standalone notion and extending them to the standard model.

We require a wrapping function to be *partially injective*. The image determines x , and only a bounded number of hash values y are consistent with a given image. For FO the first holds because the TDF is injective, and the bound is the number of keys under which the symmetric ciphertext c_2 decrypts to a valid message. This is why the symmetric scheme appears in the assumption. Under the KEM formulation of FO, a ciphertext has no component that depends on the hash value, so it admits no such bound, which is a second reason to work with hybrid encryption.

Our notion: distributional injective extractability. Coin-based extractability notions, starting with Canetti and Dakdouk [31], give the extractor the adversary’s classical coins, and so does the coin-based injective extractability notion we define. However, we show that plain coin-based extractability is incompatible with post-quantum one-wayness (Theorem 5.4), so post-quantum security needs a different formulation. Our notion generalizes the *distributional* approach that Zhandry suggested for post-quantum knowledge assumptions based on isogenies [76]. The idea is that an offline simulator, *without* the adversary’s coins, must emulate the adversary’s transcript of its interaction with the ideal inversion oracle, achieving a similar acceptance probability on a given test predicate of the transcript. The simulator may depend on the test predicate, and it does not answer inversion queries online.

The inversion oracle returns preimages of valid wrapped images, but rejects queries for images with the same preimage as the challenge. The challenge is generated by a *preimage creator*, which sees the hidden setup state and the adversary’s challenge input, and produces a hidden preimage. The preimage creator plays the role of the plaintext creator in the PA2 definition of plaintext awareness [13]. The simulator may query the challenge oracle, but it has no inversion oracle.

CPA-to-CCA lifting theorem. We prove that the FO hybrid encryption scheme is IND-CCA2-secure when G satisfies DINJEXT for the FO wrapper and one-query UCE security for strongly computationally unpredictable sources, under the stated correctness and security assumptions on the base schemes (Theorem 6.2). This notion of UCE also implies IND-CPA security of the scheme. The theorem also holds under the corresponding post-quantum assumptions.

We note that the distributional extractability relaxation we work with prevents a straightforward proof structure. Namely, a standard proof would replace the decryption oracle by a simulator once and for all, then argue about the resulting game. In our case, a simulator only reproduces the

probability that one fixed test predicate accepts, so it tells us nothing about the simulated game beyond that one test predicate. The proof instead bounds three probabilities and applies the assumption separately to each, namely the probability that the adversary guesses the challenge bit, the probability that the ideal inversion oracle answers some query differently from decryption under sk , and the probability that the adversary submits a ciphertext that shares the challenge preimage and is valid under the challenge symmetric key. Each of the three uses its own test predicate and gives its own simulator. We give the details in Section 6.2.

Modular composition theorem. A single hash function family satisfying both UCE and DINJEXT may be hard to build. To address this, we also give a composition theorem that combines a UCE hash function and a separate hash function satisfying DINJEXT for an associated wrapping function to achieve both properties together, under an additional unpredictability-preservation condition. This separates the two design tasks while isolating the conditions needed for their composition. The theorem and its proof are in Appendix A (Theorem A.4).

Toward a general class of wrappers. We would prefer to make an assumption that is not specific to FO, stated for a general class of wrapping functions rather than for the FO one. This would be similar to the way UCE [10] is stated for a class of sources, or correlation intractability [32] for a class of relations. We have not found such a general class for which DINJEXT is plausibly achievable. To guide future study, we give four counterexamples that any such general class must exclude (Section 5.5). Two of them plant an obfuscated program, one in the public setup and one in the challenge image, following the technique used against extractable one-way functions and differing-inputs obfuscation with auxiliary input [19, 48].

We leave finding such a class as the main open problem raised by our work. Settling it would say what makes FO a good wrapper, rather than leaving that as something established one scheme at a time, and it would let DINJEXT be applied to other transforms.

In this work, we do not give a standard-model construction of a hash function satisfying DINJEXT relative to the FO wrapper or any more general class. This parallels the initial situations for correlation intractability and UCE. Correlation intractability was introduced together with a proof that no hash function achieves it for the general class of sparse relations, and standard-model constructions came later for restricted classes, starting with [29]. UCE was introduced without a standard-model construction, then shown unachievable for the class of all computationally unpredictable sources assuming indistinguishability obfuscation [24], with constructions for narrower source classes following [26, 36].

Achievability in idealized models. Finally, we show that, under suitable assumptions, a random oracle satisfies 1-DINJEXT2 for every partially injective wrapping triple whose honest algorithms have no oracle access, and that a quantum random oracle satisfies a QROM version of the definition, in which the simulator answers some of the random-oracle queries, for the FO wrapping triple (Section 7). Furthermore, both results hold even relative to an *oblivious sampling oracle*, which returns wrapper images whose preimages the adversary does not know. This contrasts with our standard-model impossibility of post-quantum coin-based extractability, which uses oblivious sampling (Theorem 5.4). While it may seem backward to show our notion is achievable in the ROM and QROM, one reason this is worthwhile is that some proposed hash function notions

for random-oracle instantiation cannot be constructed in the ROM, while some others can. For example, extremely lossy functions [73] cannot [45], while UCEs can [10].

1.5 Related Work

Prior instantiability results on FO and OAEP. Kiltz, O’Neill, and Smith [63] gave an IND-CPA instantiation of OAEP, using t -wise independent hashing together with lossiness of the underlying trapdoor permutation. Bellare, Hoang, and Keelveedhi [10], in the work that introduced UCE, gave standard-model IND-CPA instantiations of the Encrypt-with-Hash mechanism [7] and of OAEP from UCE-secure hash function families. Murphy, O’Neill, and Zaheri [65] gave the first standard-model IND-CCA2 instantiation of a *modified* variant of the FO transform, for lossy PKE [11], combining indistinguishability obfuscation, extremely lossy functions [73], and multi-bit auxiliary-input point obfuscation to build G .

Negative and adjacent UCE results. Brzuska, Farshim, and Mittelbach [24] showed, assuming iO, that UCE for computationally unpredictable sources is unachievable, and subsequently [25] that the H step of FO with an IND-CPA scheme cannot be instantiated in general. Goyal, Koppula, and Waters [49] weakened the assumptions behind such barriers. Zhandry [75] recovered them in the augmented random oracle model (AROM) and also gave a counterexample to CCA security of the full FO transform, even with lossy public-key encryption and perfectly secure symmetric encryption [75, Theorem 5.1]. Both hash functions are random oracles in his counterexample.

Positive UCE constructions exist for split sources from iO and point obfuscation [26] and for strongly unpredictable sources from the lattice-based assumptions of Chen and Mao [36]. Matsuda and Hanaoka [64] show that UCE security of the derandomization hash H and IND-CPA security of the base scheme do not by themselves give IND-CPA security of the original FO transform.

Hash-function abstractions. Proposed hash-function abstractions include correlation intractability [32, 29, 28], extractability [30, 31], preimage awareness [40], and UCE [10]. Dedicated standard-model constructions have enabled instantiations of full-domain hash signatures [55, 73], Fiat-Shamir [60, 28], and OAEP [63, 10, 65].

Plaintext and preimage awareness. Plaintext awareness asks that an adversary producing a ciphertext already know the plaintext, formalized by an extractor that recovers it [15, 8]. Our notion asks the same of wrapper images and their preimages, and the correspondence is close at the level of the games. The preimage creator in 1-INJEXT2 plays the role of the plaintext creator in the PA2 definition of Bellare and Palacio [13], and our challenge oracle plays the role of their encryption oracle. There are two salient differences, however. Plaintext awareness is a property of an encryption scheme, whereas ours is a property of a hash function relative to a wrapping function. In PA, as in our coin-based 1-INJEXT2, extractors take the adversary’s coins, which is the type of extractability Theorem 5.4 rules out against quantum adversaries. Our distributional notion DINJEXT avoids it. Preimage awareness [40] is the hash-function counterpart to PA, but it is still idealized, whereas our notion is defined in the standard model.

Observability and QROM extraction. Ananth and Bhaskar [5], building on earlier criticism by Nielsen [67], identified observability as an artificial feature of the ROM and introduced non-

observing reductions. Alamati, Maram, and Masny [2] extend this line to the post-quantum setting with the non-observable QROM. QROM security of FO was first proved by Targhi and Unruh [71], and later works gave tight QROM analyses [54, 58, 70, 57]. Zhandry’s compressed oracle technique [74] and the QROM-based online-extractability framework of Don, Fehr, Majenz, and Schaffner [41] provide the core of our QROM proof.

2 Preliminaries

2.1 Notation

Notation and conventions. We write λ for the security parameter, $\text{negl}(\lambda)$ for a negligible function, and QPT/PT/PPT for quantum polynomial-time, polynomial-time, and probabilistic polynomial-time. Unless explicitly called QPT, an efficient algorithm is classical polynomial-time, probabilistic when randomized and deterministic for predicates and maps. Thus setup samplers, preimage creators, scheme algorithms, and UCE sources are classical PPT algorithms unless stated otherwise. All algorithms, games, and advantage functions are implicitly parameterized by λ ; we write, *e.g.*, $\text{Adv}(\mathcal{A})$ rather than $\text{Adv}(\mathcal{A}, \lambda)$. For a finite set S , we write $x \leftarrow \$ S$ for uniform sampling from S . We write $[n] = \{1, \dots, n\}$. For a string s , $|s|$ denotes its length in bits, s_i denotes its i -th bit, and $s|_{[i..j]}$ denotes its restriction from the i -th bit to the j -th bit for any $1 \leq i < j \leq |s|$. For a function f , we write $\text{Im}[f] := \{f(x) : x \in \text{dom}(f)\}$ for its image set. For sets A and B , we write $\text{Funcs}(A, B)$ for the set of all functions from A to B , so that $G \leftarrow \$ \text{Funcs}(\{0, 1\}^*, \{0, 1\}^\ell)$ samples a random oracle with ℓ -bit outputs. For a randomized algorithm $\mathcal{A}$, we write $y \leftarrow \$ \mathcal{A}(\dots)$ for executing it on a fresh coin tape on the elided inputs, assigning the result to y . We write $[\mathcal{A}(\dots)]$ for the support of $\mathcal{A}$ ’s output distribution on input the elided inputs.

Statistical distance. For distributions P, Q on a finite set, their statistical distance is $\text{SD}(P, Q) \stackrel{\text{def}}{=} \frac{1}{2} \sum_z |P(z) - Q(z)|$.

2.2 Encryption Schemes and Key Encapsulation

We use both public-key and symmetric-key encryption. This subsection fixes their syntax and recalls the standard security notions we need. Our positive result also uses two properties of symmetric encryption that are not standard; they are in Section 2.4.

Definition 2.1 (Public-key encryption). A public-key encryption (PKE) scheme is a tuple $\text{PKE} = (\text{PKE.Kg}, \text{PKE.Enc}, \text{PKE.Dec})$ of PPT algorithms with message space MsgSp and coin space CoinSp . Key generation $\text{PKE.Kg}(1^\lambda)$ outputs a key pair (pk, sk) ; $\text{PKE.Enc}(pk, m; y)$ on input $m \in \text{MsgSp}$ and coins $y \in \text{CoinSp}$ outputs a ciphertext c ; $\text{PKE.Dec}(sk, c)$ outputs a message m or $\perp$.

A PKE scheme is *deterministic* if its coin space is the singleton $\{\varepsilon\}$.

Definition 2.2 (Symmetric-key encryption). A symmetric-key encryption (SE) scheme is a tuple $\text{SE} = (\text{SE.Kg}, \text{SE.Enc}, \text{SE.Dec})$ of PPT algorithms with message space MsgSp_{SE} and coin space $\text{CoinSp}_{\text{SE}}$. The key space is $\{0, 1\}^{\ell_K}$, and key generation $\text{SE.Kg}(1^\lambda)$ outputs a uniform key $K \leftarrow \$ \{0, 1\}^{\ell_K}$. Encryption $\text{SE.Enc}(K, m; \rho)$ on input $m \in \text{MsgSp}_{\text{SE}}$ and coins $\rho \in \text{CoinSp}_{\text{SE}}$ outputs a ciphertext. Decryption $\text{SE.Dec}(K, c)$ is deterministic and outputs $m \in \text{MsgSp}_{\text{SE}}$ or $\perp$. Correctness requires that for all $K \in [\text{SE.Kg}(1^\lambda)]$, messages $m \in \text{MsgSp}_{\text{SE}}$, and coins $\rho \in \text{CoinSp}_{\text{SE}}$, $\text{SE.Dec}(K, \text{SE.Enc}(K, m; \rho)) = m$.

We use the standard notions of IND-CPA and IND-CCA2 security for PKE, and IND-CPA security for symmetric encryption, as games in which $\mathcal{A}$ submits equal-length messages (m_0, m_1) , receives an encryption c^* of m_b for uniform $b \leftarrow \{0, 1\}$, and outputs a bit b' ; the advantage is $|2 \Pr[b' = b] - 1|$. For IND-CCA2 the adversary has a decryption oracle that rejects c^* . We equivalently use the left-or-right-oracle form in which the adversary makes at most one challenge query. The post-quantum variants allow QPT adversaries, whose oracle queries, including decryption queries, remain classical. For symmetric encryption, the game above is the one-time form of IND-CPA security, because the adversary has no encryption oracle. This form suffices for us, because in FO each symmetric key is used to encrypt a single message.

Definition 2.3 (Ciphertext integrity (INT-CTXT) [12]). Let $\text{SE} = (\text{SE.Kg}, \text{SE.Enc}, \text{SE.Dec})$ be a symmetric encryption scheme and $\mathcal{A}$ a PPT adversary with oracle access to an encryption oracle Enc and a verification oracle Vfy . The INT-CTXT advantage of $\mathcal{A}$ against SE is $\text{Adv}_{\text{SE}}^{\text{INT-CTXT}}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[\text{INT-CTXT}_{\text{SE}}^{\mathcal{A}} \Rightarrow 1]$, where the game is defined in Figure 1. We say SE is INT-CTXT-secure if $\text{Adv}_{\text{SE}}^{\text{INT-CTXT}}(\mathcal{A})$ is negligible for every PPT $\mathcal{A}$, and post-quantum INT-CTXT-secure if the same holds for every QPT adversary.

<u>Game $\text{INT-CTXT}_{\text{SE}}^{\mathcal{A}}$</u>	<u>$\text{Enc}(m)$</u>
$K \leftarrow \text{SE.Kg}(1^\lambda)$	$c \leftarrow \text{SE.Enc}(K, m)$
$C \leftarrow \emptyset; \text{won} \leftarrow 0$	$C \leftarrow C \cup \{c\}$
$\mathcal{A}^{\text{Enc}, \text{Vfy}}(1^\lambda)$	Return c
Return won	<u>$\text{Vfy}(c)$</u>
	$b \leftarrow [c \notin C \wedge \text{SE.Dec}(K, c) \neq \perp]$
	If $b = 1$: won $\leftarrow 1$
	Return b

Figure 1: The INT-CTXT game [12].

Remark 2.4. [Multi-verification-query INT-CTXT] The formulation in Definition 2.3 permits the adversary to make many verification queries (rather than outputting a single end-of-game forgery), which is the standard multi-query INT-CTXT of Bellare–Namprempre [12]. We use this multi-query formulation directly in our reductions.

Correctness notions for PKE. Following Hofheinz, Hövelmanns, and Kiltz [54], we say PKE is δ -correct if $\Pr[\text{CORR}_{\text{PKE}}^{\mathcal{A}} \Rightarrow 1] \leq \delta$ for every (unbounded) adversary $\mathcal{A}$, where the CORR game is defined in Figure 2. When $\delta = 0$ we say PKE is *perfectly correct*.

<u>Game $\text{CORR}_{\text{PKE}}^{\mathcal{A}}$</u>
$(pk, sk) \leftarrow \text{PKE.Kg}(1^\lambda)$
$m \leftarrow \mathcal{A}(pk, sk) \ // \ \mathcal{A} \text{ unbounded}$
$y \leftarrow \text{CoinSp}$
$m' \leftarrow \text{PKE.Dec}(sk, \text{PKE.Enc}(pk, m; y))$
Return $(m' \neq m)$

Figure 2: The δ -correctness game of [54].

Key encapsulation mechanisms. We use the standard KEM syntax.

Definition 2.5 (Key encapsulation mechanism). A key encapsulation mechanism (KEM) is a tuple $\text{KEM} = (\text{KEM.Kg}, \text{KEM.Encaps}, \text{KEM.Decaps})$ of PPT algorithms. Key generation $\text{KEM.Kg}(1^\lambda)$ outputs a key pair (pk, sk) ; $\text{KEM.Encaps}(pk)$ outputs a ciphertext c together with a session key $K \in \{0, 1\}^{\ell_K}$; $\text{KEM.Decaps}(sk, c)$ outputs a key K or $\perp$. We say KEM is δ -correct if

$$\Pr[\text{KEM.Decaps}(sk, c) \neq K] \leq \delta,$$

where $(pk, sk) \leftarrow \$ \text{KEM.Kg}(1^\lambda)$, $(c, K) \leftarrow \$ \text{KEM.Encaps}(pk)$. When $\delta = 0$ we say KEM is perfectly correct.

Definition 2.6 (KEM security). Let KEM be a KEM. The KEM-CCA2 game $\text{KEM-CCA2}_{\text{KEM}}^{\mathcal{A}}$ is defined in Figure 3. The KEM-CPA game is the same game with no access to the decapsulation oracle. The KEM-CCA2 advantage of $\mathcal{A}$ against KEM is

$$\text{Adv}_{\text{KEM}}^{\text{kem-cca2}}(\mathcal{A}) \stackrel{\text{def}}{=} |2 \Pr[\text{KEM-CCA2}_{\text{KEM}}^{\mathcal{A}} \Rightarrow 1] - 1|.$$

The KEM-CPA advantage $\text{Adv}_{\text{KEM}}^{\text{kem-cpa}}(\mathcal{A})$ is defined analogously. We say KEM is KEM-CCA2-secure if $\text{Adv}_{\text{KEM}}^{\text{kem-cca2}}(\mathcal{A})$ is negligible for every PPT $\mathcal{A}$, and post-quantum KEM-CCA2-secure if the same holds for every QPT adversary. We say KEM is KEM-CPA-secure if $\text{Adv}_{\text{KEM}}^{\text{kem-cpa}}(\mathcal{A})$ is negligible for every PPT $\mathcal{A}$, and post-quantum KEM-CPA-secure if the same holds for every QPT adversary.

Game $\text{KEM-CCA2}_{\text{KEM}}^{\mathcal{A}}$	$O_{\text{dec}}(c)$
$b \leftarrow \$ \{0, 1\}$	If $c = c^*$: return $\perp$
$(pk, sk) \leftarrow \$ \text{KEM.Kg}(1^\lambda)$	Return $\text{KEM.Decaps}(sk, c)$
$(c^*, K_1^*) \leftarrow \$ \text{KEM.Encaps}(pk)$	
$K_0^* \leftarrow \$ \{0, 1\}^{\ell_K}$	
$b' \leftarrow \$ \mathcal{A}^{O_{\text{dec}}}(1^\lambda, pk, c^*, K_b^*)$	
Return $(b = b')$	

Figure 3: KEM-CCA2 game for a KEM.

2.3 Trapdoor Functions

Injective trapdoor functions. An injective trapdoor function (TDF) is a tuple $F = (F.\text{Kg}, F.\text{Eval}, F.\text{Inv})$ of PPT algorithms with domain $F.\text{Dom} = F.\text{Dom}(\lambda)$ and the following syntax. Key generation $F.\text{Kg}(1^\lambda)$ outputs a key pair (pk, sk) . Evaluation $F.\text{Eval}(pk, x)$ on $x \in F.\text{Dom}$ is deterministic and outputs $y \in \{0, 1\}^*$. Inversion $F.\text{Inv}(sk, y)$ is deterministic and outputs $x \in F.\text{Dom}$ or $\perp$. We require that $F.\text{Eval}(pk, \cdot)$ is injective for every public key in the support of $F.\text{Kg}(1^\lambda)$. We allow inversion errors and state the required correctness separately. We say that F is *perfectly correct* if $F.\text{Inv}(sk, F.\text{Eval}(pk, x)) = x$ for every $(pk, sk) \in [F.\text{Kg}(1^\lambda)]$ and every $x \in F.\text{Dom}$.

Injectivity concerns $F.\text{Eval}$: each image determines a unique preimage. Correctness concerns the specified algorithm $F.\text{Inv}$. Perfect correctness implies injectivity, since two distinct preimages of the same image could not both be recovered. The converse need not hold: $F.\text{Inv}$ may fail even when $F.\text{Eval}$ is injective.

When the key pk is clear from context, we abbreviate the image of the TDF under pk as $\text{Im}[F] \stackrel{\text{def}}{=} \text{Im}[F.\text{Eval}(pk, \cdot)]$.

Definition 2.7 (One-wayness). Let F be an injective TDF. The one-wayness advantage of an adversary $\mathcal{A}$ is $\text{Adv}_F^{\text{ow}}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[\mathcal{A}(pk, y) = x]$, where $(pk, sk) \leftarrow \$ F.\text{Kg}(1^\lambda)$, $x \leftarrow \$ F.\text{Dom}$, $y \leftarrow F.\text{Eval}(pk, x)$. We say F is one-way if $\text{Adv}_F^{\text{ow}}(\mathcal{A})$ is negligible for every PPT $\mathcal{A}$, and post-quantum one-way if the same holds for every QPT adversary.

Definition 2.8 (δ -adaptive correctness for TDFs). Let F be an injective TDF. The adaptive-correctness advantage of an adversary $\mathcal{A}$ is

$$\text{Adv}_F^{\text{adapt-corr}}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr \left[\begin{array}{l} (pk, sk) \leftarrow \$ F.\text{Kg}(1^\lambda), x \leftarrow \mathcal{A}(1^\lambda, pk, sk) : \\ x \in F.\text{Dom} \wedge F.\text{Inv}(sk, F.\text{Eval}(pk, x)) \neq x \end{array} \right].$$

We say F is δ -adaptively correct if this advantage is at most δ for every PPT adversary $\mathcal{A}$, and post-quantum δ -adaptively correct if the same holds for every QPT adversary.

If we view F as a PKE scheme with empty coins, Definition 2.8 is the CORR game of Figure 2 restricted to efficient adversaries, so δ -correctness implies δ -adaptive correctness, and in particular perfect correctness implies 0-adaptive correctness. The adversary in Definition 2.8 receives both keys and chooses x , so δ bounds its ability to find an inversion failure. A small failure probability for uniformly sampled x alone does not establish this property. Our positive FO theorem allows negligible adaptive correctness error while retaining exact injectivity.

Discussion of adaptive correctness. Adaptive correctness (Definition 2.8) gives a PPT adversary the key pair (pk, sk) and asks it to output a domain point at which inversion fails. This is related to the *find-failing-ciphertext* (FFC) game of Bindel, Hamburg, Hövelmanns, Hülsing, and Persichetti [18], which gives the adversary only pk and asks for a list containing a failing ciphertext, and to the single-shot formulations behind the tighter CCA analyses of [70, 59, 50]. Adaptive correctness and FFC security are incomparable in general. An adaptive-correctness adversary receives sk , which an FFC adversary does not. Conversely, an FFC adversary outputs only a failing ciphertext, whereas an adaptive-correctness adversary must output its preimage, and injectivity makes that preimage unique but gives no efficient way to recover it.

When F comes from the H step, a failing point x is exactly a failing message-coin pair $(x, H(K_H, x))$ for the underlying PKE. For deployed schemes the evidence that adaptive correctness holds comes from the *failure-boosting* cryptanalysis literature [38, 39]. Those analyses do not establish it formally, since our adversary also receives sk and must produce an explicit failing point.

Almost-always lossy trapdoor functions. We use a one-sided almost-always variant of the Peikert–Waters notion of lossy trapdoor functions [68]: injectivity and inversion remain exact in the injective mode, but the image bound in the lossy mode may fail with negligible probability.

Definition 2.9 (Almost-always lossy TDF). An almost-always lossy trapdoor-function family

$$\text{LT} = (\text{IKg}, \text{LKg}, \text{Eval}, \text{Inv})$$

consists of PPT algorithms, has domain $\{0, 1\}^{n(\lambda)}$ and range $\{0, 1\}^{q(\lambda)}$, and has the following properties.

1. Exact injectivity and inversion. The algorithm $\text{IKg}(1^\lambda)$ outputs (ek, td) such that $u \mapsto \text{Eval}(ek, u)$ is injective and

$$\text{Inv}(td, \text{Eval}(ek, u)) = u$$

for every $u \in \{0, 1\}^{n(\lambda)}$.

2. Mode indistinguishability. *The evaluation-key ensembles*

$$\{ek : (ek, td) \leftarrow \text{IKg}(1^\lambda)\}_\lambda \quad \text{and} \quad \{ek : ek \leftarrow \text{LKg}(1^\lambda)\}_\lambda$$

are computationally indistinguishable.

3. Almost-always lossiness. *There are functions ℓ_{loss} and β_{loss} such that*

$$\Pr_{ek \leftarrow \text{LKg}(1^\lambda)} \left[|\text{Im}(\text{Eval}(ek, \cdot))| \leq 2^{n(\lambda) - \ell_{\text{loss}}(\lambda)} \right] \geq 1 - \beta_{\text{loss}}(\lambda).$$

We call an evaluation key satisfying the displayed image bound good.

This relaxation matches what our proofs need. We apply lossiness only to an honestly generated lossy key, condition on the key being good, and charge the bad-key event as an additive β_{loss} . For the counterexample, we choose $\ell_{\text{loss}} = \kappa + \lambda$ and $\beta_{\text{loss}} = \text{negl}(\lambda)$; exact lossiness is the special case $\beta_{\text{loss}} = 0$.

Instantiation from LWE. Peikert and Waters [69, Section 6] construct lossy trapdoor functions from standard decisional LWE, with polynomial modulus and inverse error rate, for any constant lossiness rate below one. With rate $3/4$ (their Theorem 6.6) and a lattice dimension $s \geq \lambda$ large enough that the input length $n = \Theta(s \log s)$ is at least $2(\kappa + \lambda)$, the lossiness is at least $\frac{3}{4}n \geq \kappa + \lambda$. In injective mode, their construction inverts correctly only with overwhelming probability over key generation. We make inversion exact by testing the sampled error matrix in both the injective and the lossy key generation and replacing it by the zero matrix if the test fails. The test requires, in every column, that the sum of the positive entries and the absolute sum of the negative entries are both less than $Q/(4P)$, where P and Q are the plaintext and LWE moduli, which is the error bound of their Lemma 6.3. It fails with probability at most $w2^{-n}$ for w columns, so the key distributions are statistically close. The inversion and image-size arguments of their Theorem 6.4 then hold for every key, which gives Definition 2.9 with $\beta_{\text{loss}} = 0$. Quantum hardness of LWE gives the post-quantum variant.

2.4 Two Properties of Symmetric Encryption

Key-ambiguity. We introduce a notion of *key-ambiguity* for symmetric encryption, which bounds how many keys can make a length-bounded ciphertext valid, meaning it decrypts to something other than $\perp$.

Definition 2.10 (Key-ambiguity). *Let SE be a symmetric encryption scheme with key length ℓ_K . For an integer $L \geq 0$, the L -bounded key-ambiguity of SE is*

$$\Gamma_{\text{SE}}(\lambda, L) \stackrel{\text{def}}{=} \max_{c \in \{0,1\}^{\leq L}} |\{K \in \{0,1\}^{\ell_K} : \text{SE.Dec}(K, c) \neq \perp\}|,$$

and we call $\Gamma_{\text{SE}}(\lambda, L)/2^{\ell_K}$ the corresponding key-ambiguity ratio.

We also write Γ_{SE} for the function $t \mapsto \Gamma_{\text{SE}}(\lambda, t)$, which is nondecreasing in t because the maximum is over $\{0,1\}^{\leq t}$.

For a fixed L , the case $\Gamma_{\text{SE}}(\lambda, L) = 1$ is a perfect, length- L form of *key-robustness* [1]; ordinary key-robustness asks that no ciphertext, of any length, is valid under two different keys, and its

computational form asks only that such a ciphertext be hard to find. It is also closely related to *key-commitment* [9], which asks that the encryption algorithm itself does not produce a ciphertext valid under some second key. However, deployed AEAD schemes such as AES-GCM and ChaCha20-Poly1305 satisfy *neither* key-robustness nor key-commitment [3]. This motivates our relaxation to $\Gamma_{\text{SE}}(\lambda, L) \ll 2^{\ell_K}$. Lemma 2.11 below shows that INT-CTXT security against non-uniform adversaries makes this ratio negligible, which is why the relaxation is plausible for deployed AEAD schemes.

INT-CTXT bounds the key-ambiguity ratio. The next lemma bounds the key-ambiguity ratio by the INT-CTXT advantage of a non-uniform adversary.

Lemma 2.11 (INT-CTXT bounds key-ambiguity). *Let SE be a symmetric encryption scheme with key length ℓ_K and uniform key generation over this key space. For every polynomial L , there is a non-uniform INT-CTXT adversary $\mathcal{B}_L$ such that*

$$\Gamma_{\text{SE}}(\lambda, L)/2^{\ell_K} \leq \text{Adv}_{\text{SE}}^{\text{INT-CTXT}}(\mathcal{B}_L).$$

In particular, if SE is INT-CTXT-secure against non-uniform adversaries, then $\Gamma_{\text{SE}}(\lambda, L)/2^{\ell_K}$ is negligible for every polynomial L .

Proof. For each λ , let $c_\lambda \in \{0,1\}^{\leq L(\lambda)}$ attain the maximum in the definition of $\Gamma_{\text{SE}}(\lambda, L)$. The non-uniform adversary $\mathcal{B}_L$ hardcodes c_λ , makes no Enc queries, and queries $\text{Vfy}(c_\lambda)$. Then its INT-CTXT advantage is exactly $|\{K : \text{SE.Dec}(K, c_\lambda) \neq \perp\}|/2^{\ell_K} = \Gamma_{\text{SE}}(\lambda, L)/2^{\ell_K}$. $\square$

Standard INT-CTXT is often stated against non-uniform adversaries. In this case, Lemma 2.11 says that $\Gamma_{\text{SE}}(\lambda, L)/2^{\ell_K}$ is negligible for every polynomial L . However, strictly speaking, this is an asymptotic use of non-uniformity; concrete claims about, say, 128-bit security require some care [16]. We omit such claims here.

Image-closedness. We need decryption to reject every string outside the image of encryption.

Definition 2.12 (Image-closedness). *We say that a symmetric encryption scheme SE is image-closed if for every key K and every string c ,*

$$\text{SE.Dec}(K, c) \neq \perp \implies \exists m \in \text{MsgSp}_{\text{SE}}, \rho \in \text{CoinSp}_{\text{SE}} \text{ such that } \text{SE.Enc}(K, m; \rho) = c.$$

Combined with correctness, image-closedness gives, for every K ,

$$\{c : \text{SE.Dec}(K, c) \neq \perp\} = \text{Im}(\text{SE.Enc}(K, \cdot; \cdot)).$$

Thus, SE.Dec is an efficient membership test for $\text{Im}(\text{SE.Enc})$. Image-closedness is a weakening of the rigidity notion of [17]. Standard nonce-based AEAD schemes used in practice satisfy it, since a valid ciphertext-nonce pair is one that encryption could have produced with that nonce.

2.5 Hash Functions and Universal Computational Extractors

Keyed hash function families. A *keyed hash function family* is a tuple $\text{Hash} = (\text{Hash.Gen}, \text{Hash.Eval})$ of efficient algorithms. Here $\text{Hash.Gen}(1^\lambda)$ outputs a key hk and $\text{Hash.Eval}(\text{hk}, \cdot) : \mathcal{D} \rightarrow \mathcal{R}$ is deterministic on the application-specific domain and range. When convenient, we identify the family with the induced keyed map $\text{Hash}(\text{hk}, \cdot)$, or write the family itself as a function $\text{Hash} : \mathcal{K} \times \mathcal{D} \rightarrow \mathcal{R}$.

Universal computational extractors. We recall the UCE (Universal Computational Extractor) notion of Bellare, Hoang, and Keelveedhi [10]. All sources in our results make a single hash query; we state the definition for this single-query setting and vary the source class as needed.

A *single-query UCE adversary* is a pair $\mathcal{A} = (S, D)$, where the *source* S is a stateless oracle algorithm that makes *exactly one* query x to HASH and outputs leakage L , and the *distinguisher* D , given (L, hk) , outputs a bit. A *source class* $\mathcal{S}$ is a set of single-query sources satisfying some condition; we define the source classes we use below.

Definition 2.13 ($\mathcal{S}$ -UCE security [10]). Let $\text{Hash} = (\text{Hash.Gen}, \text{Hash.Eval})$ be a keyed hash function family, $\mathcal{S}$ a class of single-query sources, and $\mathcal{A} = (S, D)$ a single-query UCE adversary with $S \in \mathcal{S}$. The UCE game $\text{UCE}_{\text{Hash}}^{\mathcal{A}}$ is defined in Figure 4. The UCE advantage of $\mathcal{A}$ against Hash is $\text{Adv}_{\text{Hash}}^{\text{uce}}(\mathcal{A}) \stackrel{\text{def}}{=} |2 \Pr[\text{UCE}_{\text{Hash}}^{\mathcal{A}} \Rightarrow 1] - 1|$. We say Hash is $\mathcal{S}$ -UCE-secure if $\text{Adv}_{\text{Hash}}^{\text{uce}}(\mathcal{A}) = \text{negl}(\lambda)$ for every $S \in \mathcal{S}$ and every PPT distinguisher D .

We say Hash is post-quantum $\mathcal{S}$ -UCE-secure if the same condition holds for QPT distinguishers. The sources we use in post-quantum statements are still classical PPT algorithms. For source classes defined through prediction, the post-quantum reading also allows QPT predictors.

Game $\text{UCE}_{\text{Hash}}^{\mathcal{A}}$	$\text{HASH}(x)$
$\beta \leftarrow_{\$} \{0, 1\}$	If $x \notin Q$:
$\text{hk} \leftarrow_{\$} \text{Hash.Gen}(1^\lambda)$	$Q \leftarrow Q \cup \{x\}$
$Q \leftarrow \emptyset, T \leftarrow \emptyset$	If $\beta = 0$: $T[x] \leftarrow_{\$} \mathcal{R}$
$L \leftarrow_{\$} S^{\text{HASH}}(1^\lambda)$	Else: $T[x] \leftarrow \text{Hash.Eval}(\text{hk}, x)$
$\beta' \leftarrow_{\$} D(L, \text{hk})$	Return $T[x]$
Return $(\beta = \beta')$	

Figure 4: Single-query UCE security game [10].

UCE source classes: cup and scup. Our results use the *strongly computationally unpredictable* (scup) class of Bellare, Hoang, and Keelveedhi [10], restricted to single-query sources; we also recall the *computationally unpredictable* (cup) class for comparison. Intuitively, a source is unpredictable if its leakage L does not let a predictor output S 's hash query; the strong variant additionally reveals the oracle's responses on those queries to the predictor. Granting the predictor strictly more information only makes S 's task harder, so $\mathcal{S}_1^{\text{scup}} \subseteq \mathcal{S}_1^{\text{cup}}$. Hence $\mathcal{S}_1^{\text{cup}}$ -UCE security is the *stronger* assumption.

Formally, both classes are defined via the *prediction game* $\text{Pred}_{S,P}^{\text{xcup}}$ of Figure 5, parameterized by a variant $\text{xcup} \in \{1\text{-cup}, 1\text{-scup}\}$. The game runs S with the same HASH oracle as the $\beta = 0$ game of the UCE definition (Figure 4): a lazy-sampled random function that records S 's queries in a set Q and its responses in a table T , and S outputs leakage L . The predictor P then receives a view view and outputs a guess x' ; the game returns 1 iff $x' \in Q$. The variant selects view : in the 1-cup variant $\text{view} = L$, so P sees only the leakage, whereas in the 1-scup variant $\text{view} = (L, T.\text{values})$, so P additionally sees the oracle's responses on S 's queries.

A single-query source S is in $\mathcal{S}_1^{\text{cup}}$ (resp. $\mathcal{S}_1^{\text{scup}}$) if for every PPT predictor P ,

$$\Pr[\text{Pred}_{S,P}^{1\text{-cup}} \Rightarrow 1] = \text{negl}(\lambda) \quad \left(\text{resp. } \Pr[\text{Pred}_{S,P}^{1\text{-scup}} \Rightarrow 1] = \text{negl}(\lambda) \right).$$

In the post-quantum case, the same definition quantifies over QPT predictors. Correspondingly, S_1^{cup} -UCE security of a keyed hash function family is a *stronger* assumption than S_1^{scup} -UCE security.

Game $\text{Pred}_{S,P}^{\text{xcup}}$	$\text{HASH}(x)$
$Q \leftarrow \emptyset, T \leftarrow \emptyset$	If $x \notin Q$:
$L \leftarrow_{\$} S^{\text{HASH}}(1^\lambda)$	$Q \leftarrow Q \cup \{x\}$
If $\text{xcup} = 1\text{-cup}$: $\text{view} \leftarrow L$	$T[x] \leftarrow_{\$} \mathcal{R}$
If $\text{xcup} = 1\text{-scup}$: $\text{view} \leftarrow (L, T.\text{values})$	Return $T[x]$
$x' \leftarrow_{\$} P(1^\lambda, \text{view})$	
Return $(x' \in Q)$	

Figure 5: Prediction game for single-query UCE sources.

2.6 Obfuscation and Leftover Hash Lemma

The two tools below are used only by the negative results of Section 4 and Appendix B.

Static-input X-pIO. Indistinguishability obfuscation is stated for deterministic circuits, and the circuit we obfuscate is randomized. We therefore use *probabilistic indistinguishability obfuscation* (pIO), in the exact-output X-pIO variant of Canetti, Lin, Tessaro, and Vaikuntanathan [33], which they build from subexponentially secure indistinguishability obfuscation and subexponentially secure puncturable PRFs. Since the obfuscated circuit is deterministic, it cannot use fresh randomness on each execution. Instead, on every input it selects one output that the original randomized circuit could have produced.

The security condition compares two randomized circuits via one execution on an input chosen in advance. The adversary first chooses an input v . Only afterward are the circuits C_0, C_1 and their correlated auxiliary information z sampled. The adversary then sees both circuits and one sample from $C_b(v)$, and tries to determine b . The order is important here. Namely, the adversary cannot tailor v to the sampled circuits; this is what “static input” means.

The parameter X bounds a fixed set of exceptional inputs on which the two circuits may differ. Outside this set, they agree for every choice of randomness. At an input selected in advance, the adversary’s distinguishing advantage is at most v/X . The factor $1/X$ allows a hybrid argument to sum over all exceptional inputs while retaining a negligible bound. We now formalize these requirements.

Definition 2.14 (Static-input X-Ind sampler [33]). *Let $1 \leq X(\Lambda) \leq 2^\Lambda$. A circuit sampler $\mathcal{D}_\Lambda$ outputs (C_0, C_1, z) , where C_0, C_1 are same-size randomized circuits of size at most Λ with the same input, randomness, and output lengths, and z is arbitrary correlated auxiliary information. For $b \in \{0, 1\}$ and a two-stage adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the static-input experiment is defined in Figure 6. Its static-input advantage is*

$$\left| \Pr[\text{Static-Input-IND}_{\mathcal{D},1}^{\mathcal{A}}(1^\Lambda) \Rightarrow 1] - \Pr[\text{Static-Input-IND}_{\mathcal{D},0}^{\mathcal{A}}(1^\Lambda) \Rightarrow 1] \right|.$$

A circuit sampler $\mathcal{D} = \{\mathcal{D}_\Lambda\}_\Lambda$ is X-Ind if, for every Λ , there is a fixed set X_Λ of size at most $X(\Lambda)$, independent of the sampled circuits, such that:

1. **Agreement outside the exceptional set:** With overwhelming probability over $(C_0, C_1, z) \leftarrow \mathcal{D}_\Lambda$, for every $v \notin \mathcal{X}_\Lambda$ and every random tape r ,

$$C_0(v; r) = C_1(v; r);$$

2. **Static-input hiding:** For every non-uniform PPT adversary $\mathcal{A}$, there is a negligible function ν such that its static-input advantage is at most

$$\frac{\nu(\Lambda)}{X(\Lambda)}.$$

Game Static-Input-IND $_{\mathcal{D}, b}^{\mathcal{A}}(1^\Lambda)$
$(v, \text{st}) \leftarrow \mathcal{A}_1(1^\Lambda)$
$(C_0, C_1, z) \leftarrow \mathcal{D}_\Lambda$
$Y \leftarrow C_b(v)$
Return $\mathcal{A}_2(\text{st}, C_0, C_1, z, v, Y)$

Figure 6: The static-input indistinguishability experiments.

The two experiments differ only in whether Y is sampled from $C_0(v)$ or $C_1(v)$. The preceding definition identifies the randomized circuit pairs to which the compiler's security applies. We now define the compiler.

Definition 2.15 (Exact-output X-pIO compiler). *Let $1 \leq X(\Lambda) \leq 2^\Lambda$. An exact-output X-pIO compiler X-pIO is a uniform PPT algorithm with three properties.*

1. **Exact output:** On input $(1^\Lambda, C)$, where C has size at most Λ , it outputs a deterministic circuit ω , and every output of ω is an actual output of C : for every v there is a random tape r_v such that

$$\omega(v) = C(v; r_v).$$

2. **Computational correctness:** For every efficiently sampled (C, z) independent of the compiler coins and every non-uniform PPT oracle distinguisher given (C, z) , oracle access to $\omega \leftarrow \text{X-pIO}(1^\Lambda, C)$ is computationally indistinguishable from oracle access to C using fresh randomness on each query, provided the queries use distinct inputs. Queries may be adaptive; the distinguisher is not given the description of ω .
3. **Obfuscation security:** For every X-Ind sampler that outputs (C_0, C_1, z) and every non-uniform PPT distinguisher, the distributions

$$(z, \text{X-pIO}(1^\Lambda, C_0)) \quad \text{and} \quad (z, \text{X-pIO}(1^\Lambda, C_1))$$

are computationally indistinguishable.

Discussion. The descriptions C_0, C_1 are given to the adversary in the static-input experiment, but are not given to the obfuscation distinguisher unless they are separately included in z . The second property follows from the correctness requirement of CLTV [33, Definition 2.1]; the third

is their static-input X -Ind pIO security. They construct a compiler satisfying these two properties from subexponentially secure indistinguishability obfuscation for deterministic circuits and subexponentially secure puncturable PRFs [33]; see also [6] for indistinguishability obfuscation. Their compiler has the form

$$\text{X-pIO}(1^\Lambda, C) = iO(1^{\Lambda'}, [v \mapsto C(v; \text{PRF}_{k_{\text{prf}}}(v))])$$

for a freshly generated PRF key k_{prf} and polynomially related Λ' . Under the standard convention that iO preserves functionality exactly, this construction also has the first property: for each v , take $r_v = \text{PRF}_{k_{\text{prf}}}(v)$. Its computational correctness follows from PRF security and correctness of deterministic iO [33, Appendix B.2]. Here “subexponentially secure” means that the distinguishing advantage of every non-uniform PPT adversary is at most $2^{-\Lambda^\epsilon}$ for some constant $\epsilon > 0$.

The single-query case of computational correctness gives the consequence used in our attack analysis. If an efficient sampler produces (C, v, z) independently of the compiler coins, then

$$(z, \text{X-pIO}(1^\Lambda, C)(v)) \approx_c (z, C(v; R)), \quad (1)$$

where R is fresh uniform randomness of the required length. The auxiliary information z may contain C and relevant trapdoors, but not the compiled object itself. Thus an experiment that uses the compiled object only for one designated evaluation may replace that evaluation by a fresh randomized execution. This is a statement about the whole experiment, and does not assert uniform output randomness after conditioning on the compiled object. Exact output alone would not imply this property.

Average conditional collision probability. For jointly distributed random variables X, Z over finite sets, define

$$\text{cp}(X | Z) \stackrel{\text{def}}{=} \sum_{z: \Pr[Z=z] > 0} \Pr[Z=z] \sum_x \Pr[X=x | Z=z]^2.$$

The inner sum is the squared ℓ_2 norm of the conditional distribution of X given $Z = z$, equivalently the probability that two independent samples from it collide, and $\text{cp}(X | Z)$ is its average over Z . We use the following conditional form of the leftover-hash lemma. It follows from the standard lemma [52] by conditioning and Jensen’s inequality.

Lemma 2.16 (Leftover-hash lemma with side information). *Let $\mathcal{H} = \{h_v : \mathcal{X} \rightarrow \{0, 1\}^m\}_{v \in \mathcal{V}}$ be a two-universal family, meaning that*

$$\Pr_{V \leftarrow \mathcal{V}} [h_V(x) = h_V(x')] \leq 2^{-m}$$

for all distinct $x, x' \in \mathcal{X}$. Let $V \leftarrow \mathcal{V}$ be independent of (X, Z) , and let $U_m \leftarrow \{0, 1\}^m$ be independent of (X, Z, V) . Then

$$\text{SD}((Z, V, h_V(X)), (Z, V, U_m)) \leq \frac{1}{2} \sqrt{2^m \text{cp}(X | Z)}.$$

Proof. For each z in the support of Z , let X_z have the conditional distribution of X given $Z = z$. Since the two distributions in the statement have the same marginal distribution on Z , their statistical distance is

$$\sum_z \Pr[Z=z] \text{SD}((V, h_V(X_z)), (V, U_m)).$$

The standard leftover-hash lemma bounds the summand corresponding to z by

$$\frac{1}{2} \sqrt{2^m \sum_x \Pr[X = x \mid Z = z]^2}.$$

It follows that the distance in the statement is at most

$$\frac{1}{2} \sqrt{2^m} \sum_z \Pr[Z = z] \sqrt{\sum_x \Pr[X = x \mid Z = z]^2}.$$

Since the square-root function is concave, Jensen's inequality bounds this by

$$\frac{1}{2} \sqrt{2^m \sum_z \Pr[Z = z] \sum_x \Pr[X = x \mid Z = z]^2} = \frac{1}{2} \sqrt{2^m \text{cp}(X \mid Z)},$$

as claimed. $\square$

3 The Fujisaki–Okamoto Transform

The Fujisaki–Okamoto (FO) transform [46] converts a public-key encryption scheme that is CPA-secure, or only one-way, into one that is CCA-secure in the random-oracle model, and it is used by Kyber and the derived ML-KEM standard [21, 66]. As described in Section 1, it hashes twice, with H deriving the encryption coins and G deriving the key. We take the H step as given and define the two forms of the G step that our results concern. Both start from an injective TDF $F = (F.\text{Kg}, F.\text{Eval}, F.\text{Inv})$ with domain $F.\text{Dom}$.

Let $G : \mathcal{K}_G \times \{0, 1\}^* \rightarrow \{0, 1\}^{\kappa(\lambda)}$ be a keyed hash function family. We write $G.\text{Kg}$ for its key generation, K_G for its key, and $G(K_G, \cdot)$ for its evaluation.

Message-only implicit rejection. Our negative result uses the KEM $U_m^\perp[F, G]$ of Hofheinz, Hövelmanns, and Kiltz [54], which derives the key from the preimage alone and returns a fallback key rather than $\perp$ on an invalid ciphertext. Key generation samples $(ek, td) \leftarrow \$ F.\text{Kg}(1^\lambda)$, independently samples $K_G \leftarrow \$ G.\text{Kg}(1^\lambda)$ and $Z \leftarrow \$ F.\text{Dom}$, and returns

$$\widehat{ek} = (ek, K_G), \quad \widehat{td} = (td, K_G, Z).$$

Encapsulation samples $u \leftarrow \$ F.\text{Dom}$ and returns $(F.\text{Eval}(ek, u), G(K_G, u))$. Decapsulation computes $u' = F.\text{Inv}(td, c)$ and returns

$$\begin{cases} G(K_G, u'), & u' \neq \perp \text{ and } F.\text{Eval}(ek, u') = c, \\ G(K_G, Z \| c), & \text{otherwise.} \end{cases}$$

Decapsulation thus inverts c and re-evaluates F on the result; when F comes from the H step, this is FO's re-encryption check.

Hybrid-encryption formulation. Our positive result concerns FO as a hybrid public-key encryption scheme, as in [65]: the key $G(K_G, x)$ derived from the preimage is used with a symmetric scheme SE to encrypt the message. This is also the form our wrapping-function treatment needs, since the first ciphertext component determines the trapdoor-function preimage and the second encrypts the message under the derived key. Let SE have key length ℓ_K , and let $G : \mathcal{K}_G \times \{0, 1\}^* \rightarrow \{0, 1\}^{\ell_K}$. The scheme $\text{FO}[\text{F}, \text{SE}, G]$ is defined in Figure 7. Decryption performs the same re-evaluation check as the KEM.

$\text{FO.Kg}(1^\lambda)$	$\text{FO.Enc}(\widehat{pk}, m)$	$\text{FO.Dec}(\widehat{sk}, (c_1, c_2))$
$(pk, sk) \leftarrow \text{F.Kg}(1^\lambda)$	$(pk, K_G) \leftarrow \widehat{pk}$	$(sk, pk, K_G) \leftarrow \widehat{sk}$
$K_G \leftarrow G.\text{Kg}(1^\lambda)$	$x \leftarrow \text{F.Dom}$	$x \leftarrow \text{F.Inv}(sk, c_1)$
$\widehat{pk} \leftarrow (pk, K_G)$	$c_1 \leftarrow \text{F.Eval}(pk, x)$	If $x = \perp$: return $\perp$
$\widehat{sk} \leftarrow (sk, pk, K_G)$	$K \leftarrow G(K_G, x)$	If $\text{F.Eval}(pk, x) \neq c_1$:
Return $(\widehat{pk}, \widehat{sk})$	$c_2 \leftarrow \text{SE.Enc}(K, m)$	return $\perp$
	Return (c_1, c_2)	$K \leftarrow G(K_G, x)$
		Return $\text{SE.Dec}(K, c_2)$

Figure 7: The hybrid FO scheme $\text{FO}[\text{F}, \text{SE}, G]$.

4 A Negative Result for $U_m^\perp$

We show that no fixed hash function G makes the KEM $U_m^\perp$ secure for every one-way injective TDF. Recall from Section 3 that $U_m^\perp$ derives the session key from the sampled preimage alone, and that on an invalid ciphertext it returns a fallback key rather than $\perp$.

Previously, Brzuska, Farshim, and Mittelbach [25] showed that the H step cannot be instantiated when the base scheme is only assumed IND-CPA secure, assuming indistinguishability obfuscation. Goyal, Koppula, and Waters reduced this to lockable obfuscation and fully homomorphic encryption, and Zhandry further reduced it to circularly secure LWE [49, 75]. Murphy, O’Neill, and Zaheri [65] showed instantiability of a modified hybrid-encryption variant. Our negative result addresses the message-only G step under ordinary one-wayness.

Theorem 4.1 (Implicit-rejection uninstantiability). *Let $G : \mathcal{K}_G \times \{0, 1\}^* \rightarrow \{0, 1\}^\kappa$ be a keyed hash function. Assume an almost-always lossy TDF (Definition 2.9) with lossiness $\ell_{\text{loss}} = \kappa + \lambda$ and negligible β_{loss} . Assume an exact-output X-pIO compiler as in Definition 2.15. Then there is a one-way, perfectly correct injective TDF F_G^* , PPT adversary $\mathcal{A}$, and a polynomial t such that*

$$\text{Adv}_{U_m^\perp[\text{F}_G^*, G]}^{\text{kem-cca2}}(\mathcal{A}) \geq \frac{1}{2t} - 2^{-\kappa(\lambda)-1} - \text{negl}(\lambda). \quad (2)$$

Canetti, Lin, Tessaro, and Vaikuntanathan construct the required compiler from subexponentially secure deterministic iO and puncturable PRFs [33]; Definition 2.15 records the needed properties. The attack is classical and uses post-challenge queries. The theorem claims ordinary one-wayness, without a validity or inversion oracle.

4.1 Building the Counterexample

Let $\text{LT} = (\text{IKg}, \text{LKg}, \text{Eval}, \text{Inv})$ be an almost-always lossy TDF (LTDF) with input length n and lossiness $\ell_{\text{loss}} = \kappa + \lambda$, where κ is the output length of G . LTDFs with these parameters can be based

on the standard LWE assumption. Let ℓ_G be the key length of G and let q be the output length of LT. Put $L_0 = q + \ell_G + \kappa + 2$ and set

$$t = 2^{\lceil \log_2(4(L_0 + \lambda + 1)) \rceil},$$

the least power of two that is at least $4(L_0 + \lambda + 1)$. We take t to be a power of two so that the domain of $F_G^\star$ defined below can be sampled uniformly with a fixed number of coin flips. The counterexample TDF $F_G^\star$ uses $2t$ LTDF key pairs $\{(ek_{j,i}, td_{j,i})\}_{j \in \{0,1\}, i \in [t]}$. Besides all $ek_{j,i}$'s, its evaluation key also contains uniformly chosen strings $a_{j,i} \leftarrow \{0,1\}^n$ and an obfuscation of a leaky circuit C_{leak} . The circuit C_{leak} can be used to break the CCA security of $U_m^L[F_G^\star, G]$.

Evaluation and inversion. The domain of $F_G^\star$ is $\text{F.Dom}_G^\star = \{0,1\} \times [t] \times \{0,1\}^n$. On input (j, i, x) , evaluation outputs $c = (j, i, \text{Eval}(ek_{j,i}, x), \langle a_{j,i}, x \rangle)$. We sometimes refer to the bit j as the tag. The inversion algorithm uses $td_{j,i}$ to recover x from the third component of c and performs a re-evaluation check. If the check passes, it outputs (j, i, x) ; otherwise it returns $\perp$.

The leaky circuit C_{leak} . Define a predicate $P_{\vec{td}_0}$ as follows. The predicate has the trapdoors $\vec{td}_0 = (td_{0,i})_{i \in [t]}$, evaluation keys, and public vectors hardcoded. On input a ciphertext c , a hash key K_G , and a candidate key K , it

1. parses $c = (j, i, y, \tau)$ and returns 0 if $j \neq 0$;
2. computes $x = \text{Inv}(td_{0,i}, y)$ and returns 0 if $x \notin \{0,1\}^n$;
3. returns 1 if and only if $\text{Eval}(ek_{0,i}, x) = y \wedge \langle a_{0,i}, x \rangle = \tau \wedge G(K_G, (0, i, x)) = K$.

We omit $\vec{td}_0$ and write P for this predicate for ease of notation. An adversary with access to P can break the CCA security of $U_m^L[F_G^\star, G]$. However, since P has part of the trapdoors hardcoded, it is hard to show that $F_G^\star$ remains one-way given an obfuscation of P . Instead, C_{leak} first computes $\beta = P(c, K_G, K)$ and outputs an encryption of β with two properties:

- an adversary with access to the decapsulation oracle for $U_m^L[F_G^\star, G]$ can use this output in the CCA attack below;
- in the one-wayness proof, the obfuscation hides β .

The description of C_{leak} is given in Figure 8 below. The complete description of $F_G^\star$ is given in Figure 9.

4.2 The Attack

The adversary in Figure 10 uses the public challenge tag to choose a test. Let $\text{Flip}(c)$ denote the string obtained by flipping the last bit of c .

Let $\Theta = (\{(ek_{j,i}, td_{j,i}, a_{j,i})\}_{j,i}, K_G, Z)$ denote the base state, excluding the independent compiler coins and ω . For fixed Θ , write

$$\bar{c}_{i,x} = \text{Flip}(F_G^\star.\text{Eval}(pk, (1, i, x)))$$

Figure 8: The circuit C_{leak} .

- **Input** (c, K_G, K)
 - **Hardcoded:** $(ek_{j,i}, a_{j,i})_{i \in [t], j \in \{0,1\}}, \vec{td}_0 = (td_{0,i})_{i \in [t]}$.
1. Compute $\beta = P(c, K_G, K)$.
 2. Choose $b_1, \dots, b_t \in \{0,1\}$ uniformly conditioned on $\bigoplus_{i=1}^t b_i = \beta$: sample the first $t-1$ bits uniformly and use the parity constraint to determine b_t .
 3. Independently choose $X_1, \dots, X_t \leftarrow_{\$} \{0,1\}^n$.
 4. For each $i \in [t]$, compute
$$\begin{aligned} Q_i &= (1, i, \text{Eval}(ek_{1,i}, X_i), b_i \oplus \langle a_{1,i}, X_i \rangle), \\ R_i &= G(K_G, (1, i, X_i)). \end{aligned} \tag{3}$$
 5. Output $((Q_i, R_i))_{i \in [t]}$.

and define

$$\begin{aligned} p_i(\Theta) &= \Pr_{X \leftarrow_{\$} \{0,1\}^n} [G(K_G, Z \| \bar{c}_{i,X}) = G(K_G, (1, i, X))], \\ S(\Theta) &= \sum_{i \in [t]} p_i(\Theta), \quad \alpha(\Theta) = \prod_{i \in [t]} (1 - p_i(\Theta)). \end{aligned} \tag{4}$$

Lemma 4.2 (Parity decoding cannot reverse the signal). *Let p_{real} and p_{rand} be the probabilities that the adversary in Figure 10 outputs 1, conditioned on the challenge having tag 0, in the real-key and random-key games, respectively. Then*

$$p_{\text{real}} - p_{\text{rand}} \geq (1 - 2^{-\kappa}) \mathbb{E}_{\Theta}[\alpha(\Theta)] - \text{negl}(\lambda).$$

Proof. By the correctness of X -pIO, we may replace the adversary's evaluation of ω by a fresh randomized execution of C_{leak} , changing each acceptance probability by a negligible amount. Indeed, its input (c^*, K_G, K^*) and the auxiliary information Θ are independent of the compiler coins; Θ suffices to simulate decapsulation, and the adversary makes no other use of ω .

Consider this modified experiment using C_{leak} and fix Θ . For $B \in \{0,1\}$, let q_B be the probability that the decoded parity is 1 when C_{leak} samples its bits with parity B . Define $E_i = [G(K_G, Z \| \bar{c}_{i,X_i}) = R_i]$. When $b_i = 0$, decapsulation returns R_i . When $b_i = 1$, Q_i is invalid and decapsulation returns the fallback key. Thus $\hat{b}_i = b_i(1 - E_i)$. The indicators E_i are independent and have probabilities $p_i(\Theta)$.

If no collision occurs, the decoded parity equals B . Otherwise, decoding retains only a proper subset of the sampled bits. Every nonempty proper subset has uniform parity under either choice of B ; the empty subset has parity zero. Consequently,

$$q_1 - q_0 = \Pr[E_i = 0 \text{ for every } i] = \alpha(\Theta).$$

$\begin{aligned} & \overline{F_G^\star.\text{Kg}(1^\lambda)} \\ & \text{For } j \in \{0, 1\}, i \in [t]: \\ & \quad (ek_{j,i}, td_{j,i}) \leftarrow \$\text{IKg}(1^\lambda) \\ & \quad a_{j,i} \leftarrow \$\{0, 1\}^n \\ & \omega \leftarrow \$\text{X-plO}(1^\lambda, C_{\text{leak}}) \\ & pk \leftarrow (\{(ek_{j,i}, a_{j,i})\}_{j,i}, \omega) \\ & sk \leftarrow \{(ek_{j,i}, td_{j,i}, a_{j,i})\}_{j,i} \\ & \text{Return } (pk, sk) \end{aligned}$	$\begin{aligned} & \overline{F_G^\star.\text{Eval}(pk, u)} \\ & \text{Parse } u = (j, i, x) \in F.\text{Dom}_G^\star \\ & y \leftarrow \text{Eval}(ek_{j,i}, x); \tau \leftarrow \langle a_{j,i}, x \rangle \\ & \text{Return } (j, i, y, \tau) \\ & \overline{F_G^\star.\text{Inv}(sk, c)} \\ & \text{Parse } c = (j, i, y, \tau); \text{ else return } \perp \\ & x := \text{Inv}(td_{j,i}, y) \\ & \text{If } x \notin \{0, 1\}^n: \text{ return } \perp \\ & \text{If } \text{Eval}(ek_{j,i}, x) \neq y: \text{ return } \perp \\ & \text{If } \langle a_{j,i}, x \rangle \neq \tau: \text{ return } \perp \\ & \text{Return } (j, i, x) \end{aligned}$
--	---

Figure 9: The counterexample TDF for message-only implicit rejection.

$\begin{aligned} & \overline{\text{Adversary } \mathcal{A}^{\mathcal{O}_{\text{dec}}}(1^\lambda, (pk, K_G), c^*, K^*)} \\ & \text{Parse the tag } j^* \text{ of } c^* \\ & \text{If } j^* = 1: T \leftarrow \mathcal{O}_{\text{dec}}(\text{Flip}(c^*)); \text{ return } [T = K^*] \\ & ((Q_i, R_i))_{i \in [t]} \leftarrow \omega(c^*, K_G, K^*) \\ & \text{For } i \in [t]: T_i \leftarrow \mathcal{O}_{\text{dec}}(Q_i); \hat{b}_i \leftarrow [T_i \neq R_i] \\ & \text{Return } [\bigoplus_{i \in [t]} \hat{b}_i = 1] \end{aligned}$

Figure 10: The CCA adversary for $U_m^L[F_G^\star, G]$.

A real candidate key makes C_{leak} choose parity 1. A uniform candidate makes it choose parity 1 with probability $\varepsilon = 2^{-\kappa}$, and parity 0 otherwise. Thus the conditional difference between acceptance probabilities is

$$q_1 - (\varepsilon q_1 + (1 - \varepsilon)q_0) = (1 - \varepsilon)\alpha(\Theta).$$

Averaging over Θ and accounting for the negligible cost of the replacement proves the claim. $\square$

Proposition 4.3 (Implicit-rejection attack advantage). *The adversary of Figure 10 satisfies Equation (2); every query differs from the challenge.*

Proof. Write $\varepsilon = 2^{-\kappa}$. For $j \in \{0, 1\}$, let a_j be the real-key acceptance probability minus the random-key acceptance probability, conditioned on challenge tag j . By Lemma 4.2,

$$a_0 \geq (1 - \varepsilon)\mathbb{E}_\Theta[\alpha(\Theta)] - \text{negl}(\lambda).$$

On tag one, the flipped challenge is invalid. For fixed Θ , its fallback answer equals the real key with probability $S(\Theta)/t$, averaged over the uniform index and preimage, and equals a uniform candidate with probability ε . Hence $a_1 = \mathbb{E}_\Theta[S(\Theta)/t] - \varepsilon$. Since the challenge tag is uniform, the total signed advantage satisfies

$$\frac{a_0 + a_1}{2} \geq \frac{1}{2}\mathbb{E}_\Theta \left[(1 - \varepsilon)\alpha(\Theta) + \frac{S(\Theta)}{t} - \varepsilon \right] - \text{negl}(\lambda). \quad (5)$$

If $\varepsilon \geq 1/t$, the claimed advantage bound is nonpositive and holds trivially. Otherwise, $t \geq 2$ implies $1 - \varepsilon \geq 1/t$. Fix Θ and abbreviate $S = S(\Theta)$ and $\alpha = \alpha(\Theta)$. The union bound gives $\alpha \geq 1 - S$.

If $S \geq 1$, then $(1 - \varepsilon)\alpha + S/t \geq 1/t$. If $S \leq 1$, then

$$(1 - \varepsilon)\alpha + \frac{S}{t} \geq (1 - \varepsilon)(1 - S) + \frac{S}{t} \geq \frac{1 - S}{t} + \frac{S}{t} = \frac{1}{t}.$$

Substituting into Equation (5) gives a signed advantage of at least $1/(2t) - \varepsilon/2 - \text{negl}(\lambda)$, which proves the claimed bound.

No query equals the challenge ciphertext. For a tag-zero challenge, exact-output correctness ensures that every Q_i returned by ω has tag one; for a tag-one challenge, the query differs in its last bit. $\square$

4.3 One-Wayness

Theorem 4.4 (One-wayness of the counterexample). *Under the assumptions of Theorem 4.1, $F_G^\star$ is one-way in the sense of Definition 2.7.*

Proof overview. The detailed reductions are in Section 4.4.4. Recovering x from $(ek, a, \text{Eval}(ek, x), \langle a, x \rangle)$ for independent uniform a, x is hard by mode indistinguishability and the lossy image bound (Lemma 4.6). The remaining task is to show that publishing ω preserves this hardness.

To this end, we aim to replace ω by an obfuscation of a dummy circuit that contains no trapdoor. Define $C_{\text{dum}}(c, K_G, K)$ to ignore c, K , set $\beta = 0$, and execute steps 2–5 of C_{leak} . It contains only $(ek_{1,i}, a_{1,i})_{i \in [t]}$ and the code of G . The key step is to show that these circuits satisfy the X -Ind condition required by X -pIO when the keys $ek_{1,i}$ are generated independently in lossy mode, with the public evaluation data and an honest tag-zero challenge as auxiliary information.

Assuming this claim, consider the following hybrid sequence for a tag-zero challenge:

$$\begin{aligned} (\text{injective } ek_{1,i}, X\text{-pIO}(C_{\text{leak}})) &\approx_c (\text{lossy } ek_{1,i}, X\text{-pIO}(C_{\text{leak}})) \\ &\approx_c (\text{lossy } ek_{1,i}, X\text{-pIO}(C_{\text{dum}})) \\ &\approx_c (\text{injective } ek_{1,i}, X\text{-pIO}(C_{\text{dum}})). \end{aligned}$$

The first and last transitions follow from mode indistinguishability, switching each key in its public and hardwired occurrences together. Neither circuit uses $td_{1,i}$. The second transition follows from the claimed X -Ind property and X -pIO security. In the final hybrid, choose the challenge index I uniformly from $[t]$ and embed the hard instance $(ek, a, \text{Eval}(ek, x), \langle a, x \rangle)$ at $(0, I)$. Since C_{dum} contains no trapdoor, the reduction can construct the entire public key without the target trapdoor. Successful inversion recovers x .

For a tag-one challenge, embed the hard instance directly at a uniformly chosen $(1, I)$ in the original experiment. The reduction can construct C_{leak} , because it uses $ek_{1,I}, a_{1,I}$ but not $td_{1,I}$. Successful inversion again recovers x . Both embeddings have the correct challenge distribution and incur no index-guessing loss. Thus, subject to the claim, both tag-conditional inversion probabilities are negligible, proving one-wayness.

It remains to establish the claimed X -Ind property. Consider the sampler that generates the tag-zero keys in injective mode, the tag-one keys independently in lossy mode, and all public vectors independently and uniformly. Its auxiliary information consists of the public evaluation data and an honest tag-zero challenge. The common well-formed input domain is

$$\mathcal{V} = (\{0, 1\} \times [t] \times \{0, 1\}^q \times \{0, 1\}) \times \{0, 1\}^{\ell_G} \times \{0, 1\}^\kappa, \quad |\mathcal{V}| = t2^{L_0}.$$

Set $P = 0$ on malformed inputs and syntactically invalid hash-key encodings, assigning hash value 0^κ to the latter. We use $\mathcal{V}$ as the exceptional set and take $X(\Lambda) = |\mathcal{V}|$, with the polynomially larger compiler parameter Λ specified in Lemma 4.11.

Fix an input $v = (c, K_G, K)$ before running the sampler. Whenever $P(v) = 0$, the two circuits agree on every random tape. Otherwise, they execute the same output procedure with parity one and parity zero, respectively. We therefore bound the distance between these two output distributions.

For fixed ek, a, K_G, i , define

$$D_b^{ek, a, K_G, i} = (\text{Eval}(ek, U), b \oplus \langle a, U \rangle, G(K_G, (1, i, U))), \quad U \leftarrow_{\$} \{0, 1\}^n,$$

and write

$$\Delta(ek, a, K_G, i) = \text{SD}(D_0^{ek, a, K_G, i}, D_1^{ek, a, K_G, i}).$$

These distributions describe one output pair, omitting its fixed tag and index. In lossy mode, revealing the image and the κ -bit reference key leaves residual lossiness $\ell_{\text{loss}} - \kappa = \lambda$. Since the reference computation does not depend on a , the conditional leftover-hash lemma gives

$$\mathbb{E}_{\substack{ek \leftarrow_{\$} \text{LKg}(1^\lambda) \\ a \leftarrow_{\$} \{0, 1\}^n}} [\Delta(ek, a, K_G, i)] \leq \rho \stackrel{\text{def}}{=} \beta_{\text{loss}} + 2^{-(\lambda-1)/2} = \text{negl}(\lambda), \quad (6)$$

uniformly for every fixed K_G, i , where β_{loss} is the probability that ek fails to be lossy; see Lemma 4.7.

Now fix $(ek_{1,i}, a_{1,i})_{i \in [t]}$ and K_G , and let $\mathcal{R}_B$ denote the output distribution of steps 2–5 with $\beta = B$. The parity product identity (Lemma 4.8) gives

$$\text{SD}(\mathcal{R}_0, \mathcal{R}_1) = \prod_{i \in [t]} \Delta(ek_{1,i}, a_{1,i}, K_G, i). \quad (7)$$

The evaluation keys and $a_{j,i}$ vectors are sampled independently, and hence

$$\mathbb{E}[\text{SD}(\mathcal{R}_0, \mathcal{R}_1)] \leq \rho^t \leq \frac{2^{-\lambda}}{|\mathcal{V}|} \quad (8)$$

for all sufficiently large λ , by our choice of t (Lemma 4.10).

Conditioned on the sampled circuit descriptions and auxiliary information, the output distance at v is either zero or $\text{SD}(\mathcal{R}_0, \mathcal{R}_1)$. Averaging consequently gives the same bound for the complete static-input views, including both circuit descriptions and the auxiliary information. Outside $\mathcal{V}$, the circuits agree on every random tape. Since Λ is polynomially related to λ , the bound is of the required form $\frac{\text{negl}(\Lambda)}{X(\Lambda)}$. This establishes the X -Ind condition and justifies the obfuscation replacement used above. $\square$

Proof of Theorem 4.1. By construction, F_G^* is injective and perfectly correct. The theorem then follows from Proposition 4.3 and Theorem 4.4. $\square$

4.4 Supporting Proofs

The four subsections below give the detailed proofs for the counterexample above. We use the notation and parameters of Section 4.1; in particular, $\ell_{\text{loss}} = \kappa + \lambda$. Write β_{loss} for the probability that a key generated by LKg fails to satisfy the lossy image bound.

4.4.1 Inversion with a Public Inner Product

For an injective-mode key pair (ek, td) and a public vector $a \in \{0, 1\}^n$, recover the encoded bit and input from (y, τ) as follows. Compute $x = \text{Inv}(td, y)$ and return $\perp$ unless $x \in \{0, 1\}^n$ and $\text{Eval}(ek, x) = y$. Otherwise return

$$(b, x) = (\tau \oplus \langle a, x \rangle, x).$$

Lemma 4.5 (Recovering the encoded bit). *For every injective key pair $(ek, td) \in [\text{IKg}(1^\lambda)]$ and every $a, x \in \{0, 1\}^n$ and $b \in \{0, 1\}$, the preceding procedure recovers (b, x) from $(\text{Eval}(ek, x), b \oplus \langle a, x \rangle)$. Consequently, for fixed (ek, a) , the map*

$$(b, x) \mapsto (\text{Eval}(ek, x), b \oplus \langle a, x \rangle)$$

is injective.

Proof. Exact inversion recovers x from $\text{Eval}(ek, x)$, and the last component recovers $b \oplus \langle a, x \rangle \oplus \langle a, x \rangle = b$. Applying this decoder to two equal encodings proves injectivity. $\square$

The one-wayness proof uses the following inversion problem. Sample

$$(ek, td) \leftarrow \text{IKg}(1^\lambda), \quad a, x \leftarrow \{0, 1\}^n, \quad (y, \tau) = (\text{Eval}(ek, x), \langle a, x \rangle).$$

An inverter receives (ek, a, y, τ) and succeeds if it recovers x . The public vector a is independent of x .

Lemma 4.6 (One-wayness with a public inner product). *Under TDF mode indistinguishability and $\beta_{\text{loss}} + 2^{1-\ell_{\text{loss}}} = \text{negl}(\lambda)$, every PPT inverter succeeds in this inversion problem with negligible probability.*

Proof. Given an evaluation key ek from the mode-indistinguishability game, a distinguisher samples

$$a, x \leftarrow \{0, 1\}^n, \quad (y, \tau) = (\text{Eval}(ek, x), \langle a, x \rangle).$$

It runs the inverter on (ek, a, y, τ) and accepts exactly when the answer is x . In injective mode its acceptance probability is the inverter's success probability.

Fix a lossy key ek satisfying the image bound and any $a \in \{0, 1\}^n$. The deterministic map

$$g_{ek,a}(x) = (\text{Eval}(ek, x), \langle a, x \rangle)$$

has image size at most $2^{n-\ell_{\text{loss}}+1}$. For every image value (y, τ) , the inverter receives the same input for all preimages in $g_{ek,a}^{-1}(y, \tau)$. Hence

$$\sum_{x \in g_{ek,a}^{-1}(y, \tau)} \Pr[\mathcal{B}(ek, a, y, \tau) = x] \leq 1.$$

Summing over the image shows that, for uniform x , even an unbounded inverter recovers x with probability at most

$$\frac{|\text{Im}(g_{ek,a})|}{2^n} \leq 2^{1-\ell_{\text{loss}}}.$$

The same argument applies to the classical output distribution of a quantum inverter. Averaging over a , and bounding success by one when the key fails to satisfy the image bound, gives the lossy-mode bound $\beta_{\text{loss}} + 2^{1-\ell_{\text{loss}}}$. Mode indistinguishability completes the proof. $\square$

4.4.2 Hiding the Encoded Bit and Parity

Each output pair (Q_i, R_i) of C_{leak} encodes b_i and includes the reference key R_i . As in Section 4.3, fix ek, a , a hash key $K_G \in \mathcal{K}_G$, and $i \in [t]$, and define

$$D_b^{ek, a, K_G, i} = (\text{Eval}(ek, U), b \oplus \langle a, U \rangle, G(K_G, (1, i, U))), \quad U \leftarrow_{\$} \{0, 1\}^n,$$

and put $\Delta(ek, a, K_G, i) = \text{SD}(D_0^{ek, a, K_G, i}, D_1^{ek, a, K_G, i})$.

Lemma 4.7 (Statistical hiding with a reference key). *For every lossy key ek satisfying the image bound and every fixed $K_G \in \mathcal{K}_G$ and $i \in [t]$,*

$$\mathbb{E}_{A \leftarrow_{\$} \{0, 1\}^n} [\Delta(ek, A, K_G, i)] \leq 2^{-(\lambda-1)/2}.$$

Consequently, uniformly over all fixed K_G and i ,

$$\mathbb{E}_{\substack{ek \leftarrow_{\$} \text{LKg}(1^\lambda) \\ A \leftarrow_{\$} \{0, 1\}^n}} [\Delta(ek, A, K_G, i)] \leq \rho(\lambda) \stackrel{\text{def}}{=} \beta_{\text{loss}}(\lambda) + 2^{-(\lambda-1)/2}.$$

The function ρ is negligible.

Proof. Fix such a key ek , and write

$$V = (Y, R) = (\text{Eval}(ek, U), G(K_G, (1, i, U))).$$

The deterministic map from U to V has image size at most $2^{n-\ell_{\text{loss}}+\kappa}$. If N_v is the size of its fiber at v , then the average conditional collision probability is

$$\text{cp}(U | V) = \sum_{v \in \text{Im}(V)} \frac{N_v}{2^n} \frac{1}{N_v} = \frac{|\text{Im}(V)|}{2^n} \leq 2^{-(\ell_{\text{loss}}-\kappa)}.$$

The uniform vector A is independent of (U, V) : K_G is fixed, and the reference computation does not depend on A . The family $x \mapsto \langle A, x \rangle$ is two-universal, so Lemma 2.16, with one output bit, gives

$$\text{SD}((A, V, \langle A, U \rangle), (A, V, U_1)) \leq \frac{1}{2} \sqrt{2^{1-(\ell_{\text{loss}}-\kappa)}},$$

where U_1 is an independent uniform bit. Flipping the last bit preserves the uniform comparison distribution. The triangle inequality therefore bounds the distance between the distributions for $b = 0$ and $b = 1$, with A included, by $2^{-(\ell_{\text{loss}}-\kappa-1)/2} = 2^{-(\lambda-1)/2}$. Since the marginal distribution of A is the same in both distributions, this distance is exactly $\mathbb{E}_A[\Delta(ek, A, K_G, i)]$.

When the image bound fails, use $\Delta \leq 1$ and average over lossy key generation. Finally, the lossiness condition $\ell_{\text{loss}} = \kappa + \lambda$ and negligible β_{loss} imply that ρ is negligible. $\square$

The bound averages over the public vector and requires K_G to be fixed before that vector is sampled. This is the order used in the X -Ind experiment. The independence of the reference computation from the public vector allows us to apply the conditional leftover-hash lemma.

Fix $(ek_{1,i}, a_{1,i})_{i \in [t]}$ and K_G . Write $\mathcal{R}_B$ for the output distribution of steps 2–5 of C_{leak} with $\beta = B$. This is the distribution obtained by choosing $(b_1, \dots, b_t)$ uniformly subject to $\bigoplus_{i \in [t]} b_i = B$, choosing $X_1, \dots, X_t \leftarrow_{\$} \{0, 1\}^n$ independently, and returning $(Q_i, R_i)_{i \in [t]}$, where

$$\begin{aligned} Q_i &= (1, i, \text{Eval}(ek_{1,i}, X_i), b_i \oplus \langle a_{1,i}, X_i \rangle), \\ R_i &= G(K_G, (1, i, X_i)). \end{aligned}$$

Its i -th coordinate is the tagged and indexed version of $D_{b_i}^{ek_{1,i}, a_{1,i}, K_G, i}$.

Lemma 4.8 (Parity product identity). *For every fixed $(ek_{1,i}, a_{1,i})_{i \in [t]}$ and K_G ,*

$$\text{SD}(\mathcal{R}_0, \mathcal{R}_1) = \prod_{i \in [t]} \Delta(ek_{1,i}, a_{1,i}, K_G, i).$$

Proof. For each i , let $D_{i,b}$ denote $D_b^{ek_{1,i}, a_{1,i}, K_G, i}$ with its fixed tag and index added. This injective encoding preserves statistical distance. Expanding the parity-conditioned mixtures gives the signed-measure identity

$$\mathcal{R}_0 - \mathcal{R}_1 = 2^{-(t-1)} \bigotimes_{i \in [t]} (D_{i,0} - D_{i,1}).$$

The L_1 norm of a tensor product of finite signed measures is the product of their L_1 norms. Consequently,

$$\text{SD}(\mathcal{R}_0, \mathcal{R}_1) = \frac{1}{2} 2^{-(t-1)} \prod_{i \in [t]} 2\Delta(ek_{1,i}, a_{1,i}, K_G, i),$$

which is the claimed identity. $\square$

Corollary 4.9 (Average parity amplification). *For every fixed K_G , if the keys $ek_{1,i}$ are generated independently in lossy mode and the vectors $a_{1,i}$ are independently uniform, then*

$$\mathbb{E}[\text{SD}(\mathcal{R}_0, \mathcal{R}_1)] \leq \rho(\lambda)^t.$$

Proof. The coordinate pairs $(ek_{1,i}, a_{1,i})$ are independent and K_G is fixed. By Lemmas 4.7 and 4.8, the displayed expectation is

$$\prod_{i \in [t]} \mathbb{E}_{ek_{1,i}, a_{1,i}} [\Delta(ek_{1,i}, a_{1,i}, K_G, i)] \leq \rho^t.$$

Each expectation already includes the probability that the sampled key fails to satisfy the lossy image bound. $\square$

Lemma 4.10 (Parameters). *With the choice of t in Section 4.1, for all sufficiently large λ , it holds that $t \geq L_0 + \log_2 t + \lambda$ and $\rho^t \leq \frac{2^{-\lambda}}{t^{2L_0}}$.*

Proof. Recall that $L_0 = q + \ell_G + \kappa + 2$ and $t = 2^{\lceil \log_2(4(L_0 + \lambda + 1)) \rceil}$. For $t \geq 4$, $\log_2 t \leq t/2$, while $L_0 + \lambda < t/4$. Hence $L_0 + \log_2 t + \lambda < t$, proving the first claim. By Lemma 4.7, eventually $\rho \leq 1/2$, and therefore $\rho^t \leq 2^{-t} \leq 2^{-\lambda}/(t^{2L_0})$. $\square$

4.4.3 The X-Ind Condition

Recall that C_{dum} ignores c , K , sets $\beta = 0$, and executes steps 2–5 of C_{leak} using the input hash key K_G . Pad C_{leak} and C_{dum} to the same size, with identical input, randomness, and output formats. As in Section 4.3, let

$$\mathcal{V} = (\{0, 1\} \times [t] \times \{0, 1\}^q \times \{0, 1\}) \times \{0, 1\}^{\ell_G} \times \{0, 1\}^{\kappa}, \quad |\mathcal{V}| = t2^{L_0}.$$

Set $P = 0$ on malformed inputs and syntactically invalid hash-key encodings, assigning hash value 0^κ to the latter.

Generate all tag-zero key pairs in injective mode, all tag-one evaluation keys independently in lossy mode, and all public vectors independently and uniformly. Sample $I \leftarrow_{\$} [t]$ and $x^\star \leftarrow_{\$} \{0, 1\}^n$, and set

$$u^\star = (0, I, x^\star), \quad c^\star = (0, I, \text{Eval}(ek_{0,I}, x^\star), \langle a_{0,I}, x^\star \rangle).$$

The circuit sampler $\mathcal{T}$ outputs $(C_{\text{leak}}, C_{\text{dum}}, z)$, with

$$z = (\{(ek_{j,i}, a_{j,i})\}_{j,i}, c^\star).$$

Thus z contains the public evaluation data and an honest tag-zero challenge. The challenge depends only on the tag-zero data, which are independent of the tag-one keys and vectors.

Lemma 4.11 (The circuit sampler is X -Ind). *There are constants a and λ_0 for which the following sampler is X -Ind. On indices $\Lambda = \lambda^a$ for an integer $\lambda \geq \lambda_0$, use the sampler above and put $X(\Lambda) = t(\lambda)2^{L_0(\lambda)}$. On every other index, output a fixed pair of identical circuits and put $X(\Lambda) = 1$. Consequently, in the hybrid with all tag-one keys lossy,*

$$(z, X\text{-pIO}(1^\Lambda, C_{\text{leak}})) \approx_c (z, X\text{-pIO}(1^\Lambda, C_{\text{dum}})).$$

Proof. Fix an index $\Lambda = \lambda^a$ with $\lambda \geq \lambda_0$, and condition on the classical input $v = (c, K_G, K)$ selected by the first stage before the sampler runs. In particular, K_G is fixed before the tag-one keys and vectors are sampled. The first stage's retained state, conditioned on v , is independent of the sampler. Condition further on the tag-zero data and auxiliary information derived from them, and on fixed public tag-one keys and vectors.

If $v \notin \mathcal{V}$ or K_G has a syntactically invalid encoding, the two circuits produce identical outputs on every tape, so their conditional distance is zero. Otherwise, if $P(c, K_G, K) = 0$, both circuits use parity zero and again produce identical outputs on every tape. If $P(c, K_G, K) = 1$, their output distributions are $\mathcal{R}_1$ and $\mathcal{R}_0$, respectively. The circuit descriptions and auxiliary data are identical side information in the two conditional views. For valid K_G , Lemma 4.8 therefore bounds their conditional statistical distance by

$$\prod_{i \in [t]} \Delta(ek_{1,i}, a_{1,i}, K_G, i).$$

Averaging over the independent tag-one keys and vectors and using Corollary 4.9 and lemma 4.10 bounds the distance between the complete static-input views by

$$\rho^t \leq \frac{2^{-\lambda}}{t2^{L_0}}.$$

This bound holds uniformly for every fixed v , so it survives averaging over the first stage. It bounds every second-stage distinguisher even though that stage receives both circuit descriptions. In the post-quantum setting, adjoining the same conditional retained quantum state preserves trace distance, giving the identical bound.

Choose a constant integer a and λ_0 such that, for every $\lambda \geq \lambda_0$, the parameter bound applies and both the padded circuit size and $\log_2 |\mathcal{V}| = L_0 + \log_2 t$ are at most λ^a . On these indices, $X(\Lambda) = |\mathcal{V}| \leq 2^\Lambda$. Take $\mathcal{V}$ as the exceptional set on these indices and the empty set on all remaining

indices. Outside $\mathcal{V}$, both circuits use parity zero and agree on every random tape; on the remaining indices, the two circuits are identical. The function

$$v(\Lambda) = 2^{-\lfloor \Lambda^{1/a} \rfloor}$$

is negligible in Λ . The static-input advantage is at most $v(\Lambda)/X(\Lambda)$ when $\Lambda = \lambda^a$ with $\lambda \geq \lambda_0$ and zero otherwise. This proves the X -Ind requirement of Definition 2.14. The claimed obfuscation indistinguishability follows from X -pIO security, where the distinguisher receives (z, ω) . $\square$

4.4.4 The One-Wayness Reduction

Detailed proof of Theorem 4.4. Let $\mathcal{A}$ be a one-wayness adversary against F_G^* . For a candidate preimage u' and challenge c , define the public test

$$\text{Win}(pk, c, u') \stackrel{\text{def}}{=} [u' \in F.\text{Dom}_G^* \wedge F_G^*.\text{Eval}(pk, u') = c].$$

In the real game, injectivity makes this equivalent to recovering the sampled preimage. The equivalence also holds in every tag-zero hybrid below, since the tag-zero keys remain injective.

We first record the advantages used in the reduction. For a mode distinguisher $\mathcal{D}$, write

$$\text{Adv}_{\text{LT}}^{\text{mode}}(\mathcal{D}) \stackrel{\text{def}}{=} \left| \Pr_{(ek, td) \leftarrow \text{IKg}(1^\lambda)} [\mathcal{D}(ek) = 1] - \Pr_{ek \leftarrow \text{LKg}(1^\lambda)} [\mathcal{D}(ek) = 1] \right|.$$

For an inverter $\mathcal{B}$ in the experiment of Lemma 4.6, write

$$\text{Adv}_{\text{LT}}^{\text{inv}}(\mathcal{B}) \stackrel{\text{def}}{=} \Pr \left[\begin{array}{l} (ek, td) \leftarrow \text{IKg}(1^\lambda), a, x \leftarrow \{0, 1\}^n, \\ y \leftarrow \text{Eval}(ek, x), \tau \leftarrow \langle a, x \rangle, \\ x' \leftarrow \mathcal{B}(ek, a, y, \tau) : x' = x \end{array} \right].$$

For the sampler $\mathcal{T}$ of Lemma 4.11, define

$$p_b(\mathcal{D}) \stackrel{\text{def}}{=} \Pr[(C_0, C_1, z) \leftarrow \mathcal{T}(1^\Lambda), \omega \leftarrow X\text{-pIO}(1^\Lambda, C_b) : \mathcal{D}(z, \omega) = 1],$$

$$\text{Adv}_{X\text{-pIO}, \mathcal{T}}^{\text{xpio}}(\mathcal{D}) \stackrel{\text{def}}{=} |p_0(\mathcal{D}) - p_1(\mathcal{D})|.$$

Here $C_0 = C_{\text{leak}}$ and $C_1 = C_{\text{dum}}$ on the parameter indices used by the construction. The distinguisher receives only (z, ω) .

The proof of Lemma 4.6 gives, for every $\mathcal{B}$, a mode distinguisher $\mathcal{D}_{\mathcal{B}}^{\text{inv}}$ such that

$$\text{Adv}_{\text{LT}}^{\text{inv}}(\mathcal{B}) \leq \text{Adv}_{\text{LT}}^{\text{mode}}(\mathcal{D}_{\mathcal{B}}^{\text{inv}}) + \beta_{\text{loss}}(\lambda) + 2^{1-\ell_{\text{loss}}(\lambda)}. \quad (9)$$

Given ek , it samples a, x uniformly, computes $y = \text{Eval}(ek, x)$ and $\tau = \langle a, x \rangle$, and accepts exactly when $\mathcal{B}(ek, a, y, \tau)$ returns x .

The tag-zero hybrids. Recall that C_{dum} ignores c, K , sets $\beta = 0$, and executes steps 2–5 of C_{leak} . It contains only $(ek_{1,i}, a_{1,i})_{i \in [t]}$ and the code of G , and hence contains no trapdoor. For $k \in \{0, \dots, t\}$, define $H_{0,k}^{\text{leak}}$ as follows. Generate every tag-zero key pair by IKg . Generate $ek_{1,i}$ in lossy mode for $i \leq k$ and in injective mode for $i > k$. Sample all $a_{j,i} \leftarrow \{0, 1\}^n$ independently, construct C_{leak} , and set

$$\omega \leftarrow \text{X-pIO}(1^\Lambda, C_{\text{leak}}), \quad pk = (\{(ek_{j,i}, a_{j,i})\}_{j,i}, \omega).$$

Sample $I \leftarrow [t]$ and $x^* \leftarrow \{0, 1\}^n$, form

$$c^* = (0, I, \text{Eval}(ek_{0,I}, x^*), \langle a_{0,I}, x^* \rangle),$$

run $u' \leftarrow \mathcal{A}(pk, c^*)$, and return $\text{Win}(pk, c^*, u')$.

Define $H_{0,k}^{\text{dum}}$ in the same way, except that $ek_{1,i}$ is injective for $i \leq k$ and lossy for $i > k$, and $\omega \leftarrow \text{X-pIO}(1^\Lambda, C_{\text{dum}})$. The endpoints of the hybrid sequence are therefore

Hybrid	Tag-one keys	Published obfuscation
$H_{0,0}^{\text{leak}}$	all injective	$\text{X-pIO}(C_{\text{leak}})$
$H_{0,t}^{\text{leak}}$	all lossy	$\text{X-pIO}(C_{\text{leak}})$
$H_{0,0}^{\text{dum}}$	all lossy	$\text{X-pIO}(C_{\text{dum}})$
$H_{0,t}^{\text{dum}}$	all injective	$\text{X-pIO}(C_{\text{dum}})$.

Let ϵ_0 be the real inversion probability conditioned on challenge tag zero. Then $\epsilon_0 = \Pr[H_{0,0}^{\text{leak}} \Rightarrow 1]$.

Switching to lossy keys. For each $k \in [t]$, a mode distinguisher $\mathcal{D}_k^{\text{leak}}$ embeds its challenge evaluation key at $(1, k)$, generates the tag-one keys with index $i < k$ in lossy mode and those with $i > k$ in injective mode, and samples all tag-zero key pairs and public vectors as above. It uses the challenge key in both its public and hardcoded occurrences, constructs C_{leak} and its obfuscation, runs $\mathcal{A}$, and returns $\text{Win}(pk, c^*, u')$. Since C_{leak} uses no tag-one trapdoor, the reduction can construct it without $td_{1,k}$. Hence

$$\left| \Pr[H_{0,k-1}^{\text{leak}} \Rightarrow 1] - \Pr[H_{0,k}^{\text{leak}} \Rightarrow 1] \right| = \text{Adv}_{\text{LT}}^{\text{mode}}(\mathcal{D}_k^{\text{leak}}). \quad (10)$$

Replacing the leaky circuit. Define $\mathcal{D}_\omega(z, \omega)$ to parse the public evaluation data and c^* from z , form $pk = (\{(ek_{j,i}, a_{j,i})\}_{j,i}, \omega)$, run $u' \leftarrow \mathcal{A}(pk, c^*)$, and return $\text{Win}(pk, c^*, u')$. On the parameter indices where $\mathcal{T}$ outputs identical circuits, it returns zero. The sampler $\mathcal{T}$ gives precisely the two endpoint distributions, so

$$\left| \Pr[H_{0,t}^{\text{leak}} \Rightarrow 1] - \Pr[H_{0,0}^{\text{dum}} \Rightarrow 1] \right| = \text{Adv}_{\text{X-pIO}, \mathcal{T}}^{\text{xpio}}(\mathcal{D}_\omega). \quad (11)$$

This advantage is negligible by Lemma 4.11 and X-pIO security. The reduction gives $\mathcal{A}$ the entire obfuscation, allowing arbitrary polynomial-time use of it.

Switching back to injective keys. For each $k \in [t]$, a mode distinguisher $\mathcal{D}_k^{\text{dum}}$ embeds its challenge key at $(1, k)$, generates the tag-one keys with index $i < k$ in injective mode and those with $i > k$ in lossy mode, and completes the hybrid with C_{dum} . It uses the challenge key in its public and hardcoded occurrences together. Since C_{dum} contains no trapdoor, this gives

$$\left| \Pr[H_{0,k-1}^{\text{dum}} \Rightarrow 1] - \Pr[H_{0,k}^{\text{dum}} \Rightarrow 1] \right| = \text{Adv}_{\text{LT}}^{\text{mode}}(\mathcal{D}_k^{\text{dum}}). \quad (12)$$

Inverting a tag-zero challenge. The inverter $\mathcal{B}_0$ receives (ek^*, a^*, y^*, τ^*) , where $y^* = \text{Eval}(ek^*, x^*)$ and $\tau^* = \langle a^*, x^* \rangle$. It samples $I \leftarrow_{\$} [t]$, embeds (ek^*, a^*) at $(0, I)$, and generates all other key pairs and vectors honestly. It constructs C_{dum} and its obfuscation without $td_{0,I}$, since the dummy circuit contains only tag-one public data. It gives $\mathcal{A}$ the resulting public key and $c^* = (0, I, y^*, \tau^*)$.

If the returned preimage passes Win, its tag and index force it to have the form $(0, I, x')$, and injectivity gives $x' = x^*$. The reduction returns x' in this case and $\perp$ otherwise. This exactly simulates $H_{0,t}^{\text{dum}}$. Since the reduction samples the challenge index itself, it incurs no index-guessing loss:

$$\Pr[H_{0,t}^{\text{dum}} \Rightarrow 1] = \text{Adv}_{\text{LT}}^{\text{inv}}(\mathcal{B}_0). \quad (13)$$

Combining these transitions gives

$$\begin{aligned} \epsilon_0 \leq & \sum_{k=1}^t \text{Adv}_{\text{LT}}^{\text{mode}}(\mathcal{D}_k^{\text{leak}}) + \sum_{k=1}^t \text{Adv}_{\text{LT}}^{\text{mode}}(\mathcal{D}_k^{\text{dum}}) \\ & + \text{Adv}_{\text{X-pIO}, \mathcal{T}}^{\text{xpio}}(\mathcal{D}_\omega) + \text{Adv}_{\text{LT}}^{\text{inv}}(\mathcal{B}_0). \end{aligned} \quad (14)$$

Inverting a tag-one challenge. Let ϵ_1 be the real inversion probability conditioned on challenge tag one. Given (ek^*, a^*, y^*, τ^*) , the inverter $\mathcal{B}_1$ samples $I \leftarrow_{\$} [t]$ and embeds (ek^*, a^*) at $(1, I)$. It generates all other key pairs and public vectors honestly. Since C_{leak} uses no tag-one trapdoor, the reduction can construct the leaky circuit and its obfuscation without $td_{1,I}$. It gives $\mathcal{A}$ the resulting public key and $c^* = (1, I, y^*, \tau^*)$. A preimage passing Win has the form $(1, I, x')$; injectivity again gives $x' = x^*$. The reduction returns x' on success and $\perp$ otherwise. This exactly simulates the real game conditioned on challenge tag one, including the published obfuscation. Thus, again without an index-guessing loss,

$$\epsilon_1 = \text{Adv}_{\text{LT}}^{\text{inv}}(\mathcal{B}_1). \quad (15)$$

The challenge tag is uniform, so $\text{Adv}_{F_G^*}^{\text{ow}}(\mathcal{A}) = \frac{\epsilon_0 + \epsilon_1}{2}$. Combining Equation (14) and Equation (15) gives

$$\begin{aligned} \text{Adv}_{F_G^*}^{\text{ow}}(\mathcal{A}) \leq & \frac{1}{2} \left(\sum_{k=1}^t \text{Adv}_{\text{LT}}^{\text{mode}}(\mathcal{D}_k^{\text{leak}}) + \sum_{k=1}^t \text{Adv}_{\text{LT}}^{\text{mode}}(\mathcal{D}_k^{\text{dum}}) \right. \\ & \left. + \text{Adv}_{\text{X-pIO}, \mathcal{T}}^{\text{xpio}}(\mathcal{D}_\omega) + \text{Adv}_{\text{LT}}^{\text{inv}}(\mathcal{B}_0) + \text{Adv}_{\text{LT}}^{\text{inv}}(\mathcal{B}_1) \right). \end{aligned} \quad (16)$$

Applying Equation (9) to both inverters gives

$$\begin{aligned} \text{Adv}_{F_G^*}^{\text{ow}}(\mathcal{A}) \leq & \frac{1}{2} \left(\sum_{k=1}^t \text{Adv}_{\text{LT}}^{\text{mode}}(\mathcal{D}_k^{\text{leak}}) + \sum_{k=1}^t \text{Adv}_{\text{LT}}^{\text{mode}}(\mathcal{D}_k^{\text{dum}}) \right. \\ & + \text{Adv}_{\text{LT}}^{\text{mode}}(\mathcal{D}_{\mathcal{B}_0}^{\text{inv}}) + \text{Adv}_{\text{LT}}^{\text{mode}}(\mathcal{D}_{\mathcal{B}_1}^{\text{inv}}) \\ & \left. + \text{Adv}_{\text{X-pIO}, \mathcal{T}}^{\text{xpio}}(\mathcal{D}_\omega) \right) + \beta_{\text{loss}}(\lambda) + 2^{1-\ell_{\text{loss}}(\lambda)}. \end{aligned} \quad (17)$$

□

5 Injective Extractability

We define the extractability-style assumption for a general keyed hash function family. The starting point is plain extractability, due to Canetti and Dakdouk [31], where an extractor recovers a preimage of the adversary’s output from the adversary’s coins. Plain extractability does not apply to FO, so we recast it relative to a partially injective *wrapping function*, which combines a preimage, the hash of that preimage, and further data into an image. This abstracts the QROM technique of Don, Fehr, Majenz, and Schaffner [41] and ports it to the standard model. In Section 5.1 we give the resulting notion, which we call *injective extractability* (INJEXT).

However, in Section 5.2, we show that plain coin-based extractability is incompatible with post-quantum one-wayness, so in Section 5.3 we recast the notion in a distributional form, *distributional injective extractability* (DINJEXT), generalizing Zhandry’s approach to post-quantum isogeny-based knowledge assumptions [76]. In Section 5.4 we place it in a broader hierarchy, following [34].

Plain extractability. The notion generalizes earlier knowledge-of-exponent assumptions [37, 51]. An adversary that outputs a point y in the image of f_k must “know” a preimage, in the sense that an extractor given the adversary’s coins can recover one.

Definition 5.1 (EXT). *An efficiently computable injective function family $f = \{f_k : \{0, 1\}^n \rightarrow \{0, 1\}^m\}_{k \in \mathcal{K}}$ is extractable (EXT) if for every PPT (resp. QPT) adversary $\mathcal{A}$ there exists a PPT (resp. QPT) extractor $\mathcal{E}$ such that*

$$\Pr \left[\begin{array}{l} k \leftarrow \mathcal{K}; r \leftarrow \{0, 1\}^{q_{\mathcal{A}}(\lambda)}; y \leftarrow \mathcal{A}(k; r) : \\ y \in \text{Im}(f_k) \wedge f_k(\mathcal{E}(k, r, y)) \neq y \end{array} \right] \leq \text{negl}(\lambda).$$

Here $q_{\mathcal{A}}$ bounds the length of the coins that $\mathcal{A}$ uses, and we interpret $f_k(x)$ as $\perp$ for $x \notin \{0, 1\}^n$.

Note that the notion is *coin-based*: in addition to the key and output, the extractor receives the adversary’s classical coins, but no other private execution state.

From plain EXT to injective extraction. Two features of FO make Definition 5.1 inapplicable to instantiability. The hash function G may be compressing, and the ciphertext the adversary submits to its decryption oracle contains no hash value. Extraction is therefore defined relative to a wrapping function, and we begin with the structural condition such a function must satisfy.

Definition 5.2 (Wrapping function, Γ -partial injectivity). *A wrapping function is a family $\mathcal{G} = \{g_p\}_{p \in \mathcal{P}}$ of maps*

$$g_p : X \times \{0, 1\}^\ell \times \mathcal{W} \times \mathcal{R} \longrightarrow \{0, 1\}^*,$$

where $\mathcal{P} = \mathcal{P}(\lambda)$ is a parameter set, $X \subseteq \{0, 1\}^{\leq q(\lambda)}$ is a preimage domain for a polynomial q , $\mathcal{W}$ is an auxiliary-input space, and $\mathcal{R}$ is a coin space. The consistency predicate of g_p is defined on all bit strings by

$$C_p(x, y, z) = 1 \iff x \in X \wedge y \in \{0, 1\}^\ell \wedge \exists w \in \mathcal{W}, \rho \in \mathcal{R} : g_p(x, y, w; \rho) = z.$$

Both g_p and C_p are required to be efficiently computable for every $p \in \mathcal{P}$. Let $\Gamma = \Gamma(\lambda, t)$ be a function of the security parameter and an image length t . The family $\mathcal{G}$ is Γ -partially injective (Γ -PI) if the following hold.

1. **x -injectivity.** *For every $p \in \mathcal{P}$, if $C_p(x, y, z) = C_p(x', y', z) = 1$ then $x = x'$.*

2. *y-ambiguity*. For every $p \in \mathcal{P}$, $x \in \mathcal{X}$, and $z \in \{0, 1\}^*$,

$$|\{y \in \{0, 1\}^\ell : C_p(x, y, z) = 1\}| \leq \Gamma(\lambda, |z|).$$

In the FO wrapping function of Section 6.1, Γ bounds the number of keys that decrypt the symmetric component of a ciphertext. So $\Gamma > 1$ there unless each ciphertext decrypts under at most one key, a perfect form of key commitment [53, 43, 9].

Wrapping triple. Every notion in this section is defined for a *wrapping triple* $(\mathcal{G}, \text{Param}, P)$, and the rest of the paper refers to triples by that name. The first component is a wrapping function, defined in Definition 5.2. We define the other two here.

A *setup sampler* for $\mathcal{G}$ is an efficient algorithm Param that, on input 1^λ , outputs a pair (p, aux) with $p \in \mathcal{P}$. The parameter p selects the wrapping function g_p and is public. The auxiliary output aux is hidden state. For example, p may be a public key and aux the corresponding secret key. The setup sampler is not given the hash key, so (p, aux) is independent of hk .

A *preimage creator* for $(\mathcal{G}, \text{Hash})$ is an efficient algorithm P with a challenge-input space $\mathcal{W}'$, and membership in $\mathcal{W}'$ must be efficiently decidable. On input $(\text{hk}, p, \text{aux}, w')$ with $w' \in \mathcal{W}'$, it outputs wrapping-function inputs $(x, w, \rho) \in \mathcal{X} \times \mathcal{W} \times \mathcal{R}$. When $\mathcal{W}'$ is not specified, it is $\{0, 1\}^*$. The preimage creator is what the games below run to produce an honestly generated image.

5.1 Coin-Based Injective Extractability

Injective extractability compares the execution of two games with the same two-stage adversary $\mathcal{A} = (\mathcal{A}_{\text{run}}, \mathcal{A}_{\text{out}})$. In the ideal inversion game, the run stage submits wrapped images to an ideal inverter, which answers with the true preimages, which are unique by x -injectivity. In the extraction game, that oracle is replaced by an efficient extractor given the adversary's coin tape, which is what makes the notion coin-based. Both games also give the run stage a challenge oracle, which runs P on an input in $\mathcal{W}'$ to produce one honestly generated image. The preimage x^* of that image goes to neither the adversary nor the extractor. We require that for every adversary there is an extractor making the two games accept with negligibly different probabilities. Acceptance is decided by the output predicate, which receives the hidden setup state aux . The extractor does not, so it has to answer without knowing what the predicate will check.

Definition 5.3 (Injective extractability). Let $\mathcal{G} = \{g_p\}_{p \in \mathcal{P}}$ be a Γ -PI wrapping function (Definition 5.2). Let $\text{Hash} = (\text{Hash.Gen}, \text{Hash.Eval})$ be a keyed hash function family with $\text{Hash.Eval}(\text{hk}, \cdot) : \{0, 1\}^* \rightarrow \{0, 1\}^\ell$ for every hash key hk . Let Param be an efficient setup sampler for $\mathcal{G}$ and let P be a preimage creator for $(\mathcal{G}, \text{Hash})$. The extraction game $\text{Ext-1-INJEXT2}_{\text{Hash}, \mathcal{G}}^{\mathcal{A}, \mathcal{E}}$ and the ideal inversion game $\text{Inv-1-INJEXT2}_{\text{Hash}, \mathcal{G}}^{\mathcal{A}}$ are defined in Figure 12. The advantage of an adversary $\mathcal{A}$ with extractor $\mathcal{E}$ is

$$\text{Adv}_{\text{Hash}, \mathcal{G}, \text{Param}, P}^{\text{1-INJEXT2}}(\mathcal{A}, \mathcal{E}) \stackrel{\text{def}}{=} \left| \Pr[\text{Ext-1-INJEXT2}_{\text{Hash}, \mathcal{G}}^{\mathcal{A}, \mathcal{E}} \Rightarrow 1] - \Pr[\text{Inv-1-INJEXT2}_{\text{Hash}, \mathcal{G}}^{\mathcal{A}} \Rightarrow 1] \right|.$$

We say Hash is $\text{1-INJEXT2}[\Gamma]$ for $(\mathcal{G}, \text{Param}, P)$ if for every PPT adversary $\mathcal{A}$ there exists a PPT extractor $\mathcal{E}$ such that $\text{Adv}_{\text{Hash}, \mathcal{G}, \text{Param}, P}^{\text{1-INJEXT2}}(\mathcal{A}, \mathcal{E})$ is negligible. We write $\text{Adv}_{\text{Hash}, \mathcal{G}, P}^{\text{1-INJEXT2}}$ when Param is clear.

The prefix 1 in 1-INJEXT2, and in 1-DINJEXT2 below, indicates that the challenge oracle answers only one query.

The extractor may depend on both stages of $\mathcal{A}$, including its output predicate. It receives the list L_{chal} of challenge images but not their preimages, just as the PA2 extractor of Bellare and Palacio [13] receives the ciphertexts returned by the encryption oracle but not their plaintexts. Weaker variants, with no challenge oracle and possibly a single inversion query, are defined in Section 5.4.

Same- x -preimage mauling queries. The extractor is not expected to find the challenge preimage x^* , which need not appear in the adversary’s execution. When $\mathcal{G}$ has non-trivial auxiliary inputs or wrapping coins, the adversary may be able to produce a different image $z' \neq z^*$ with the same preimage x^* . Answering such a query would reveal x^* , which the extractor cannot match because x^* is hidden from it, so by definition O_{inv} returns $\perp$ on such a query. In applications where same- x -preimage mauling is an issue, matching the output of O_{inv} on such queries must be argued per application.

Achievability in the ROM and QROM. In Section 7, we show that a random oracle satisfies the distributional form of this notion (Section 5.3) for every partially injective wrapping triple whose honest algorithms have no oracle access. We also prove the corresponding result for the FO wrapping triple under a QROM version of the definition, in which the simulator answers some of the random-oracle queries. In Section 5.5 and Appendix B, we show that, if injective one-way functions exist, no standard-model hash function family satisfies the notion for every wrapping triple.

5.2 Impossibility of Post-Quantum Coin-Based Extractability

Using the quantum oblivious-sampling technique of Zhandry [76, Section 1.3], we show that plain coin-based extractability (Definition 5.1) is incompatible with post-quantum one-wayness for injective function families. The intuition is that after a quantum adversary creates an image by measuring a superposition, the classical coins may be independent of the hidden preimage. An extractor that only sees those coins is then being asked to invert a one-way function. We state the theorem for plain extractability, since that is where the argument is cleanest, but nothing in it uses the structure of f beyond one-wayness, and we expect the same obstruction to apply to any coin-based variant of extractability, 1-INJEXT2 included.

Theorem 5.4 (Post-quantum impossibility of plain EXT). *Let $f = \{f_k : \{0, 1\}^n \rightarrow \{0, 1\}^m\}_{k \in \mathcal{K}}$ be an efficiently computable injective keyed function family. If f is one-way against QPT adversaries, then f is not EXT (Definition 5.1) with respect to QPT adversaries and extractors.*

Discussion. The same statement holds for the stronger variant of EXT against QPT adversaries in which the extractor is also given the private quantum state the adversary retains at termination. Discarded registers are excluded from this state, which is the right convention here because an adversary may always discard a register, and this one discards exactly the register holding the preimage. Furthermore, the theorem concerns extraction in the standard model, where the extractor receives the key, the output, and the adversary’s classical coins, but no other private execution state. It says nothing about idealized models, where the extractor has more to work

with, and indeed we obtain positive results there. A random oracle satisfies 1-DINJEXT2 for every partially injective wrapping triple whose honest algorithms have no oracle access (Proposition 7.3), and a quantum random oracle satisfies a version with an oracle-simulation interface for the FO wrapping triple (Theorem 7.4). The ROM simulator uses the adversary's query log and the QROM simulator implements the oracle away from the challenge preimage, and neither is available to a standard-model extractor.

Proof. Assume towards contradiction that f is EXT with respect to QPT adversaries and QPT extractors. We construct a QPT adversary $\mathcal{A}$ (Figure 11) that produces elements of $\text{Im}(f_k)$ using no classical randomness at all.

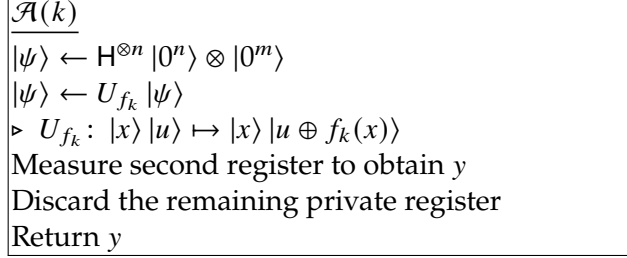


Figure 11: **QPT adversary $\mathcal{A}$** for Theorem 5.4.

The adversary discards its remaining private register before returning. Thus, under the stronger extractor interface described above, the extractor receives only a fixed empty final state; we suppress this harmless extra input below. Since $\mathcal{A}$ uses only $H^{\otimes n}$, U_{f_k} , and a computational-basis measurement—none of which depend on the coin string r —the output distribution is entirely determined by the quantum measurement:

$$\Pr[\mathcal{A}(k) \Rightarrow y'] = \Pr_{x \leftarrow \{0,1\}^n} [f_k(x) = y'] \quad \text{for all } k \in \mathcal{K}, y' \in \{0,1\}^m.$$

Since f is EXT against QPT adversaries with QPT extractors, there exists a QPT extractor $\mathcal{E}$ such that $f_k(\mathcal{E}(k, r, y)) = y$ with overwhelming probability when $r \leftarrow \{0,1\}^{q_{\mathcal{A}}(\lambda)}$ and $y \leftarrow \mathcal{A}(k; r)$. Define a QPT inverter $\mathcal{B}$ that, on input (k, y) , samples $r \leftarrow \{0,1\}^{q_{\mathcal{A}}(\lambda)}$ and outputs $\mathcal{E}(k, r, y)$. We claim $\mathcal{B}$ breaks post-quantum one-wayness of f . Write $\mathbf{D}_{\text{ow}}$ and $\mathbf{D}_{\mathcal{A}}$ for the joint distributions

$$\begin{aligned} \mathbf{D}_{\text{ow}} &: k \leftarrow \mathcal{K}, x \leftarrow \{0,1\}^n, y \leftarrow f_k(x), r \leftarrow \{0,1\}^{q_{\mathcal{A}}(\lambda)}, \\ \mathbf{D}_{\mathcal{A}} &: k \leftarrow \mathcal{K}, r \leftarrow \{0,1\}^{q_{\mathcal{A}}(\lambda)}, y \leftarrow \mathcal{A}(k; r). \end{aligned}$$

These two distributions are identical. The marginal of k agrees; in both, r is uniform and independent of (k, y) , in $\mathbf{D}_{\mathcal{A}}$ because $\mathcal{A}$ ignores r ; and by the display above the conditional law of y given k is that of $f_k(x)$ for uniform x in both. Therefore:

$$\text{Adv}_f^{\text{OW}}(\mathcal{B}) = \Pr_{\mathbf{D}_{\text{ow}}} [f_k(\mathcal{E}(k, r, y)) = y] = \Pr_{\mathbf{D}_{\mathcal{A}}} [f_k(\mathcal{E}(k, r, y)) = y] \geq 1 - \text{negl}(\lambda),$$

where the inequality follows from the assumption that f is EXT. This contradicts the post-quantum one-wayness of f . $\square$

Game Ext-1-INJEXT2$_{\text{Hash}, \mathcal{G}}^{\mathcal{A}, \mathcal{E}}$ $hk \leftarrow \\$ \text{Hash.Gen}(1^\lambda)$ $(p, aux) \leftarrow \\$ \text{Param}(1^\lambda)$ $r \leftarrow \\$ \{0, 1\}^{q_{\mathcal{A}}}$ $L_{\text{chal}} \leftarrow []; I_{\text{chal}} \leftarrow \emptyset$ $q_{\text{chal}} \leftarrow 0$ $st_{\mathcal{E}} \leftarrow r // \text{coins}$ $v \leftarrow \\$ \mathcal{A}_{\text{run}}^{O_{\text{chal}}, O_{\text{ext}}}(1^\lambda, hk, p; r)$ Return $\mathcal{A}_{\text{out}}(hk, p, aux, v)$ $O_{\text{ext}}(z)$ If $z \in L_{\text{chal}}$: return $\perp$ $(x, st_{\mathcal{E}}) \leftarrow \mathcal{E}(hk, p, z, L_{\text{chal}}, st_{\mathcal{E}})$ Return $x // x \text{ may be } \perp$ Game Sim-1-DINJEXT2$_{\text{Hash}, \mathcal{G}}^{\mathcal{B}, \mathcal{R}}$ $hk \leftarrow \\$ \text{Hash.Gen}(1^\lambda)$ $(p, aux) \leftarrow \\$ \text{Param}(1^\lambda)$ $L_{\text{chal}} \leftarrow []; I_{\text{chal}} \leftarrow \emptyset$ $q_{\text{chal}} \leftarrow 0$ $v \leftarrow \\$ \mathcal{R}^{O_{\text{chal}}}(1^\lambda, hk, p)$ Return $\mathcal{B}_{\text{out}}(hk, p, aux, v)$	Game Inv-1-INJEXT2$_{\text{Hash}, \mathcal{G}}^{\mathcal{A}}$ // = Inv-1-DINJEXT2, without the coin tape $hk \leftarrow \\$ \text{Hash.Gen}(1^\lambda)$ $(p, aux) \leftarrow \\$ \text{Param}(1^\lambda)$ $r \leftarrow \\$ \{0, 1\}^{q_{\mathcal{A}}}$ $L_{\text{chal}} \leftarrow []; I_{\text{chal}} \leftarrow \emptyset$ $q_{\text{chal}} \leftarrow 0$ $v \leftarrow \\$ \mathcal{A}_{\text{run}}^{O_{\text{chal}}, O_{\text{inv}}}(1^\lambda, hk, p; r)$ Return $\mathcal{A}_{\text{out}}(hk, p, aux, v)$ $O_{\text{inv}}(z)$ If $z \in L_{\text{chal}}$: return $\perp // \text{challenge}$ If $\nexists x \in X$ with $C_p(x, \text{Hash.Eval}(hk, x), z) = 1$: return $\perp$ Let x be the unique x-preimage of $z // \text{by } x\text{-inj.}$ If $\exists (x^*, z^*) \in I_{\text{chal}}$ with $x = x^*$ and $z \neq z^*$: return $\perp // \text{mauling}$ Return x $O_{\text{chal}}(w')$ If $w' \notin \mathcal{W}'$: return $\perp$ $q_{\text{chal}} \leftarrow q_{\text{chal}} + 1$ If $q_{\text{chal}} > 1$: return $\perp$ $(x, w, \rho) \leftarrow \\$ P(hk, p, aux, w')$ $z \leftarrow g_p(x, \text{Hash.Eval}(hk, x), w; \rho)$ $I_{\text{chal}} \leftarrow I_{\text{chal}} \cup \{(x, z)\}; \text{append } z \text{ to } L_{\text{chal}}$ Return z
--	---

Figure 12: Games for 1-INJEXT2[Γ] (extraction and ideal inversion) and for 1-DINJEXT2[Γ] (simulation and ideal inversion, the latter without the coin tape). The challenge oracle is shared by all of them.

5.3 Distributional Injective Extractability

Since we want our FO theorem to hold against quantum adversaries, we replace the extractor by an offline simulator that never sees the adversary's coins. The simulator is asked only to reproduce the acceptance probability of the output predicate, which is all the FO proof uses. This follows Zhandry's approach to post-quantum knowledge assumptions [76], and it lowers the requirement already in the classical setting, where coin-based injective extractability implies its distributional counterpart (Proposition 5.8). The basic version of the notion generalizes Zhandry's post-quantum knowledge assumptions in two ways: it considers an arbitrary hash function family and quantifies over all test predicates of the transcript.

The wrapping triple is as in Section 5. The ideal inversion game for 1-DINJEXT2 is that of Definition 5.3 with the adversary's coin tape removed, and the randomness of a QPT adversary is not captured by a classical coin tape anyway. In the post-quantum case, $\mathcal{B}_{\text{run}}$ and $\mathcal{B}_{\text{out}}$ may be QPT, but the calls of $\mathcal{B}_{\text{run}}$ to O_{chal} and O_{inv} are classical.

Definition 5.5 (Distributional injective extractability). Let $\mathcal{G} = \{g_p\}_{p \in \mathcal{P}}$ be a Γ -PI wrapping function (Definition 5.2). Let $\text{Hash} = (\text{Hash.Gen}, \text{Hash.Eval})$ be a keyed hash function family with $\text{Hash.Eval}(hk, \cdot) : \{0, 1\}^* \rightarrow \{0, 1\}^\ell$ for every hash key hk . Let Param be an efficient setup sampler for $\mathcal{G}$

and let P be a preimage creator for $(\mathcal{G}, \text{Hash})$. The simulation game $\text{Sim-1-DINJEXT2}_{\text{Hash}, \mathcal{G}}^{\mathcal{B}, \mathcal{R}}$ and the ideal inversion game $\text{Inv-1-DINJEXT2}_{\text{Hash}, \mathcal{G}}^{\mathcal{B}}$ are defined in Figure 12. The advantage of an adversary $\mathcal{B}$ with distributional simulator $\mathcal{R}$ is

$$\text{Adv}_{\text{Hash}, \mathcal{G}, \text{Param}, P}^{\text{1-DINJEXT2}}(\mathcal{B}, \mathcal{R}) \stackrel{\text{def}}{=} \left| \Pr[\text{Inv-1-DINJEXT2}_{\text{Hash}, \mathcal{G}}^{\mathcal{B}} \Rightarrow 1] - \Pr[\text{Sim-1-DINJEXT2}_{\text{Hash}, \mathcal{G}}^{\mathcal{B}, \mathcal{R}} \Rightarrow 1] \right|.$$

We say Hash is $\text{1-DINJEXT2}[\Gamma]$ for $(\mathcal{G}, \text{Param}, P)$ if for every PPT (resp. QPT) adversary $\mathcal{B}$ there exists a PPT (resp. QPT) distributional simulator $\mathcal{R}$ such that $\text{Adv}_{\text{Hash}, \mathcal{G}, \text{Param}, P}^{\text{1-DINJEXT2}}(\mathcal{B}, \mathcal{R})$ is negligible and, if, for every λ , $q_{\text{chal}} = 0$ with probability 1 at the end of the ideal inversion game with $\mathcal{B}$, then, for every λ , $q_{\text{chal}} = 0$ with probability 1 at the end of the simulation game with $\mathcal{R}$. Here q_{chal} is the challenge-query counter of Figure 12. We write $\text{Adv}_{\text{Hash}, \mathcal{G}, P}^{\text{1-DINJEXT2}}$ when Param is clear.

Discussion. The second condition is needed because the simulator runs offline. It does not answer the adversary's queries but replaces the run stage and produces a transcript on its own, so it has to be given the challenge oracle. Without the second condition, a simulator for an adversary that never queries $\mathcal{O}_{\text{chal}}$ could query it anyway and use the reply, which depends on the hidden setup state. Its success would then say nothing about the replies to the adversary's inversion queries. The second condition rules this out, because it requires the simulator for such an adversary never to query $\mathcal{O}_{\text{chal}}$. The second condition also connects $\text{1-DINJEXT2}[\Gamma]$ to the weaker notions of Section 5.4. Because of it, $\text{1-DINJEXT2}[\Gamma]$ implies $\text{DINJEXT1}[\Gamma]$ and hence $\text{DINJEXT0}[\Gamma]$ (Proposition 5.7). Two of the counterexamples in Appendix B are stated for $\iota = 0$, and this implication is what makes them counterexamples to $\text{1-DINJEXT2}[\Gamma]$ as well.

As a point of comparison, the plaintext awareness notion of Bellare and Palacio [13] needs no such condition, because the extractor has no encryption oracle. Ebrahimi and van Wier [42] define post-quantum plaintext awareness by giving the extractor the adversary's internal quantum registers in place of its coins. Again, the extractor also has no access to the plaintext creator.

5.4 The (Distributional) Injective Extractability Hierarchies

In this subsection we put 1-INJEXT2 and 1-DINJEXT2 in separate hierarchies containing weaker notions and relate the coin-based notions to the distributional ones. These results are used later in Appendix B, where we give negative results.

Definition 5.6 (The ι hierarchy). Fix $\iota \in \{0, 1, 2\}$ and let Hash , $\mathcal{G}$, Param and P be as in Definition 5.3. Modify the games of Figure 12 as follows.

1. For $\iota = 0$, the challenge oracle is removed, and the inversion and extraction oracles answer only the first query and return $\perp$ on every later one.
2. For $\iota = 1$, the challenge oracle is removed, and the inversion and extraction oracles are unchanged.
3. For $\iota = 2$, the games are unchanged.

Write $\text{INJEXT}_\iota[\Gamma]$ and $\text{DINJEXT}_\iota[\Gamma]$ for the notions defined from the modified games as in Definition 5.3 and Definition 5.5. For $\iota = 2$ these are the notions $\text{1-INJEXT2}[\Gamma]$ and $\text{1-DINJEXT2}[\Gamma]$ of those definitions.

Proposition 5.7 (Implications among the distributional notions). *Fix Γ , Hash, and a wrapping triple $(\mathcal{G}, \text{Param}, P)$ with $\mathcal{G}$ a Γ -PI wrapping function. Against PPT adversaries, and also against QPT adversaries, $1\text{-DINJEXT2}[\Gamma]$ implies $\text{DINJEXT1}[\Gamma]$, and $\text{DINJEXT1}[\Gamma]$ implies $\text{DINJEXT0}[\Gamma]$.*

Proof. In each case we run the given adversary in the game one level up and use the simulator given there. For the first implication, a DINJEXT1 adversary $\mathcal{B}$ is a 1-DINJEXT2 adversary that never queries $\mathcal{O}_{\text{chal}}$, and its ideal inversion games for $\iota = 1$ and $\iota = 2$ are the same. By the condition on challenge queries in Definition 5.5, the 1-DINJEXT2 simulator for $\mathcal{B}$ never queries $\mathcal{O}_{\text{chal}}$ either, so it is a DINJEXT1 simulator for $\mathcal{B}$ with the same advantage. For the second implication, a DINJEXT0 adversary receives $\perp$ on every inversion query after the first, so we may assume it makes only one. It is then a DINJEXT1 adversary with the same ideal inversion game, and the simulation games for $\iota = 0$ and $\iota = 1$ are the same, since neither has an inversion oracle or a challenge oracle. $\square$

From coin-based to distributional extraction. Against classical adversaries, each coin-based notion implies the corresponding distributional notion. The idea is that the simulator samples its own coin tape, runs a version of the adversary on it, forwards its challenge queries, and answers its inversion queries with the coin-based extractor.

Proposition 5.8 (Coin-based extractability implies distributional extractability). *Fix $\iota \in \{0, 1, 2\}$, Γ , Hash, and a wrapping triple $(\mathcal{G}, \text{Param}, P)$ with $\mathcal{G}$ a Γ -PI wrapping function. For every PPT DINJEXT_ι adversary $\mathcal{B} = (\mathcal{B}_{\text{run}}, \mathcal{B}_{\text{out}})$ there is a PPT INJEXT_ι adversary $\mathcal{A}_{\mathcal{B}}$ such that, for every PPT extractor $\mathcal{E}$, there is a PPT simulator $\mathcal{R}_{\mathcal{E}}$ with*

$$\text{Adv}_{\text{Hash}, \mathcal{G}, \text{Param}, P}^{\text{DINJEXT}_\iota}(\mathcal{B}, \mathcal{R}_{\mathcal{E}}) = \text{Adv}_{\text{Hash}, \mathcal{G}, \text{Param}, P}^{\text{INJEXT}_\iota}(\mathcal{A}_{\mathcal{B}}, \mathcal{E}).$$

For $\iota = 2$, the simulator $\mathcal{R}_{\mathcal{E}}$ also meets, for $\mathcal{B}$, the condition on challenge queries in Definition 5.5. In particular, if Hash is $\text{INJEXT}_\iota[\Gamma]$ for $(\mathcal{G}, \text{Param}, P)$, then it is $\text{DINJEXT}_\iota[\Gamma]$ for $(\mathcal{G}, \text{Param}, P)$ against PPT adversaries.

Proof. Write $\text{Inv}(\cdot)$, $\text{Ext}(\cdot, \cdot)$, and $\text{Sim}(\cdot)$ for the ideal inversion, extraction, and simulation games at level ι with the indicated algorithms, and write $\Pr_{\mathcal{G}}[E]$ for the probability of an event E in a game $\mathcal{G}$. Let $\mathcal{B}^-$ be $\mathcal{B}$ with its challenge query removed: whenever $\mathcal{B}_{\text{run}}$ is about to query $\mathcal{O}_{\text{chal}}$, $\mathcal{B}^-$ continues it with $\perp$ instead. Let

$$\mathcal{A}_{\mathcal{B}} = \begin{cases} \mathcal{B}^- & \text{if } \iota = 2 \text{ and } \Pr_{\text{Inv}(\mathcal{B})}[q_{\text{chal}} \geq 1] = 0 \text{ for every } \lambda, \\ \mathcal{B} & \text{otherwise.} \end{cases}$$

Which case applies depends only on $\mathcal{B}$ and not on λ , so $\mathcal{A}_{\mathcal{B}}$ is a PPT algorithm in either case. Given an extractor $\mathcal{E}$ for $\mathcal{A}_{\mathcal{B}}$, the simulator $\mathcal{R}_{\mathcal{E}}$ samples coins r , runs the run stage of $\mathcal{A}_{\mathcal{B}}$ on them, forwards its challenge query to its own challenge oracle, and answers each inversion query as $\mathcal{O}_{\text{ext}}$ does using $\mathcal{E}$ with coin tape r . We show

$$\Pr[\text{Inv}(\mathcal{B}) \Rightarrow 1] = \Pr[\text{Inv}(\mathcal{A}_{\mathcal{B}}) \Rightarrow 1], \quad (18)$$

$$\Pr[\text{Sim}(\mathcal{R}_{\mathcal{E}}) \Rightarrow 1] = \Pr[\text{Ext}(\mathcal{A}_{\mathcal{B}}, \mathcal{E}) \Rightarrow 1], \quad (19)$$

which together give the equality of advantages. Equation (19) holds because $\mathcal{R}_{\mathcal{E}}$ runs $\mathcal{A}_{\mathcal{B}}$ exactly as the extraction game does. Equation (18) is immediate when $\mathcal{A}_{\mathcal{B}} = \mathcal{B}$. When $\mathcal{A}_{\mathcal{B}} = \mathcal{B}^-$, the

games $\text{Inv}(\mathcal{B})$ and $\text{Inv}(\mathcal{B}^-)$ proceed identically until $\mathcal{B}_{\text{run}}$ is about to query $\mathcal{O}_{\text{chal}}$ on an input in $\mathcal{W}'$, since on any other input the challenge oracle returns $\perp$ anyway. The counter q_{chal} counts exactly these queries, and

$$\Pr_{\text{Inv}(\mathcal{B})}[q_{\text{chal}} \geq 1] = 0$$

by the choice of $\mathcal{A}_{\mathcal{B}}$.

For $\iota = 2$, the condition on challenge queries in Definition 5.5 asks that $\Pr_{\text{Sim}(\mathcal{R}_{\mathcal{E}})}[q_{\text{chal}} \geq 1] = 0$ for every λ whenever $\Pr_{\text{Inv}(\mathcal{B})}[q_{\text{chal}} \geq 1] = 0$ for every λ . In that case $\mathcal{A}_{\mathcal{B}} = \mathcal{B}^-$, which never queries $\mathcal{O}_{\text{chal}}$, so $\mathcal{R}_{\mathcal{E}}$ never does either. $\square$

The adversary $\mathcal{B}^-$ is needed only because an extractor may answer an inversion query incorrectly without changing the acceptance probability, and $\mathcal{B}_{\text{run}}$ could react to an incorrect answer by querying $\mathcal{O}_{\text{chal}}$.

5.5 Negative Results and an Open Problem

Our FO theorem in Section 6 assumes a hash function family that is 1-DINJEXT2 for the FO wrapping triple Wrap_{FO} . We would prefer to assume 1-DINJEXT2 for every wrapping triple in a general class that contains Wrap_{FO} for all suitable F and SE . This would parallel the treatment of correlation intractability [32] and UCE [10]. Unfortunately, we were not able to find a general class of wrapping triples for which a 1-DINJEXT2 hash function family plausibly exists, even before requiring that the class contain Wrap_{FO} .

Negative results. As a guide for future work in the area, in Appendix B we give four counterexamples that impact what a class of such wrapping triples can plausibly contain. Each uses a different channel for “dangerous data”, namely the input (Appendix B.1), the witness (Appendix B.2), the setup (Appendix B.3), and the challenge image (Appendix B.4). In each case, the adversary manages to then form a valid image without knowing its preimage. The adversary submits this image to the ideal inverter, which returns its preimage. A simulator would have to find that preimage on its own, because the output predicate checks whether the reported preimage is correct. The setup and challenge-image counterexamples use indistinguishability obfuscation, following [19, 48], and the challenge-image one is the most involved.

Pseudorandom wrapping triples. One natural candidate condition is that the wrapping triple be *pseudorandom*, meaning that the setup parameters together with an honestly generated image, formed with a uniform string in place of the hash value, is indistinguishable from uniform. The FO wrapping triple is pseudorandom if the trapdoor function has jointly pseudorandom public keys and images and the symmetric encryption scheme has pseudorandom ciphertexts. The condition plausibly excludes the two obfuscation-based counterexamples, because they would require an obfuscator whose output is indistinguishable from uniform. Obfuscation results for restricted classes of evasive programs [27, 49, 72] do not directly provide the pseudorandom obfuscations needed here. The heuristic construction of doubly pseudorandom obfuscation [23] requires the program’s entire truth table to be pseudorandom against distinguishers running in time polynomial in the table’s length. The programs in our counterexamples do not satisfy this requirement.

However, pseudorandomness on its own is insufficient. It does not exclude our input-channel counterexample when hash keys are uniform, F is a keyless one-way permutation, and the preimage

creator samples a uniform preimage. A general class would thus also need to restrict what the wrapping function computes from its preimage. We leave such an exploration to future work.

6 A Positive Result for FO Hybrid Encryption

We give sufficient hash-function assumptions for instantiating the G step of FO as defined in Section 3, taking the trapdoor function F as given. Throughout, F is assumed to be an injective trapdoor function that is one-way and δ -adaptively correct (Definition 2.8), SE is image-closed, and the keys output by $G.Kg(1^\lambda)$ have a fixed public encoding length ℓ_G .

Our theorem *lifts* IND-CPA security to IND-CCA2 security. In other words, IND-CPA security of the transform is assumed. For FO, IND-CPA security is satisfied just assuming one-wayness of the TDF, UCE of G , and IND-CPA security of SE (Proposition 6.6). The lifting proof is *not* simply to switch the decryption oracle to the extractor and reduce to IND-CPA security of the scheme. Rather, the proof is involved; see the proof outline below.

6.1 The FO Wrapping Function

Unlike most prior work analyzing FO in the ROM/QROM and implicitly using extractability through H (e.g., [54, 41, 56]), our extractability assumption for FO is on G . We first say what the wrapping function is, then show that it is partially injective.

The wrapping function. Fix the hybrid scheme $FO[F, SE, G]$ of Section 3, and write $F.Dom = \{0, 1\}^n$ for the domain of F . Define the wrapping function $\mathcal{G}^{FO} = \{g_{pk}^{FO}\}$, with preimage domain $\mathcal{X} = F.Dom$, auxiliary-input space $\mathcal{W} = \text{MsgSp}_{SE}$, and coin space $\mathcal{R} = \text{CoinSp}_{SE}$, by

$$g_{pk}^{FO}(x, K, m; \rho) \stackrel{\text{def}}{=} (F.Eval(pk, x), SE.Enc(K, m; \rho)) \quad (20)$$

for all $x \in F.Dom$, $K \in \{0, 1\}^{\ell_K}$, $m \in \text{MsgSp}_{SE}$, and $\rho \in \text{CoinSp}_{SE}$. The setup sampler Param_{FO} samples

$$(pk, sk) \leftarrow \$ F.Kg(1^\lambda), \quad x^* \leftarrow \$ F.Dom, \quad b \leftarrow \$ \{0, 1\},$$

and returns the pair (p, aux) with $p = pk$ and $aux = ((sk, x^*), b)$. The preimage creator is

$$P_{FO}(K_G, pk, ((sk, x^*), b), (m_0, m_1)) : \quad \rho \leftarrow \$ \text{CoinSp}_{SE}; \quad \text{return } (x^*, m_b, \rho).$$

Its challenge-input space $\mathcal{W}$ consists of the pairs (m_0, m_1) of equal-length messages in MsgSp_{SE} , and the challenge oracle returns $\perp$ on any other input. We write $\text{Wrap}_{FO} = (\mathcal{G}^{FO}, \text{Param}_{FO}, P_{FO})$ for the resulting FO wrapping triple.

Proposition 6.1 (Partial injectivity of the FO wrapper). *Let F be an injective TDF and let SE be image-closed. Then $\mathcal{G}^{FO}$ is $\Gamma_{SE}\text{-PI}$ with consistency predicate*

$$\begin{aligned} C_{pk}(x, K, (c_1, c_2)) = 1 & \iff x \in F.Dom \wedge K \in \{0, 1\}^{\ell_K} \\ & \wedge c_1 = F.Eval(pk, x) \wedge SE.Dec(K, c_2) \neq \perp. \end{aligned}$$

Proof. The consistency predicate is clearly efficient. By correctness and image-closedness of SE, the consistency predicate accepts exactly when (c_1, c_2) is in the image of $g_{pk}^{\text{FO}}(x, K, \cdot; \cdot)$. Injectivity of $F.\text{Eval}(pk, \cdot)$ ensures x -injectivity. For $z = (c_1, c_2)$ the consistent keys are those with $\text{SE}.\text{Dec}(K, c_2) \neq \perp$. Since c_2 has length at most $|z|$, Definition 2.10 bounds their number by $\Gamma_{\text{SE}}(\lambda, |z|)$. $\square$

Note that the need for the injectivity assumption on F comes from the x -injectivity requirement of our extractability framework, not from the FO construction itself. Treating noninjective F would require an ambiguity- or list-based extension of 1-DINJEXT2, which we leave for future work. In particular, whether the TDF derived from standardized ML-KEM via the H step is injective and adaptively correct would require separate justification that we do not address here—its negligible decapsulation-failure probabilities [66] do not by themselves imply these properties.

The UCE source. For our analysis, define the single-query UCE source S_F by

$$\begin{aligned} S_F^{\text{HASH}}(1^\lambda) : \quad & (pk, sk) \leftarrow \$F.\text{Kg}(1^\lambda); \quad x \leftarrow \$F.\text{Dom}; \\ & c_1 \leftarrow F.\text{Eval}(pk, x); \quad K \leftarrow \text{HASH}(x); \\ & \text{return } (pk, c_1, K). \end{aligned}$$

If F is one-way, this source is strongly computationally unpredictable, namely $S_F \in \mathcal{S}_1^{\text{scup}}$. Indeed, a strong predictor for this source receives the leakage (pk, c_1, K) together with the hash answer, which is K itself, and must output the query x . An inverter for F therefore samples K on its own, runs the predictor on (pk, c_1, K) , and recovers x whenever the predictor succeeds.

Hash function families that are $\mathcal{S}_1^{\text{scup}}$ -UCE-secure, and hence UCE-secure for S_F , have been built from lattice assumptions. Chen and Mao [36] construct one from LWE with subexponential hardness and private-coin evasive LWE for specific samplers, together with the existence of point obfuscation with auxiliary input in NC^1 .

6.2 The Fujisaki–Okamoto CPA-to-CCA Lifting Theorem

Our main theorem says that if the hash function G used in the G step is 1-DINJEXT2 for the FO wrapping triple, then IND-CPA security of $\text{FO}[F, \text{SE}, G]$ implies IND-CCA2 security. The remaining assumptions are standard. The trapdoor function F must be adaptively correct, the symmetric scheme SE must have ciphertext integrity, and G must also be UCE-secure for the single source S_F defined above. We show separately that IND-CPA security itself follows from one-wayness of F , UCE security of G , and IND-CPA security of SE (Proposition 6.6). The proof outline after the theorem explains where each assumption is used.

Theorem 6.2 (Instantiability of FO). *Let F be a δ -adaptively correct injective TDF, let SE be an image-closed, INT-CTXT-secure symmetric encryption scheme, and let G be $\{S_F\}$ -UCE-secure and 1-DINJEXT2 $[\Gamma_{\text{SE}}]$ for Wrap_{FO} . If δ is negligible and $\text{FO}[F, \text{SE}, G]$ is IND-CPA-secure, then it is IND-CCA2-secure.*

Concretely, for every PPT IND-CCA2 adversary $\mathcal{A}$ there are PPT 1-DINJEXT2 $[\Gamma_{\text{SE}}]$ adversaries $\mathcal{B}_1$, $\mathcal{B}_1^{\text{corr}}$, and $\mathcal{B}_1^{\text{maul}}$ such that, for all PPT simulators $\mathcal{R}_1$, $\mathcal{R}_1^{\text{corr}}$, and $\mathcal{R}_1^{\text{maul}}$, there are a PPT UCE adversary $\mathcal{B}_2$

with source S_F , a PPT IND-CPA adversary $\mathcal{B}_3$, and a PPT INT-CTXT adversary $\mathcal{B}_4$ with

$$\begin{aligned} \mathbf{Adv}_{\text{FO}}^{\text{ind-cca2}}(\mathcal{A}) &\leq 2(\Delta_1 + \Delta_1^{\text{corr}} + \Delta_1^{\text{maul}}) + 2\delta + 2\mathbf{Adv}_G^{\text{uce}}(\mathcal{B}_2) \\ &\quad + \mathbf{Adv}_{\text{FO}}^{\text{ind-cpa}}(\mathcal{B}_3) + 2\mathbf{Adv}_{\text{SE}}^{\text{INT-CTXT}}(\mathcal{B}_4), \end{aligned}$$

where $\Delta_1 = \mathbf{Adv}_{G, \text{Wrap}_{\text{FO}}}^{1\text{-DINJEXT2}[\Gamma_{\text{SE}}]}(\mathcal{B}_1, \mathcal{R}_1)$, and Δ_1^{corr} and Δ_1^{maul} are defined analogously.

Both statements also hold against QPT adversaries, with QPT in place of PPT throughout and the post-quantum versions of all assumptions.

The UCE assumption concerns only the source S_F of Section 6.1. It follows from S_1^{scup} -UCE security of G when F is one-way, since S_F is then in S_1^{scup} .

The bound Γ_{SE} is the y -ambiguity parameter in the 1-DINJEXT2 assumption. If SE is INT-CTXT-secure against non-uniform adversaries, its key-ambiguity ratio $\Gamma_{\text{SE}}(\lambda, L)/2^{\ell_K}$ is negligible for every polynomial L by Lemma 2.11. The INT-CTXT assumption is used separately to rule out same- x -preimage mauling of the challenge ciphertext.

In Appendix A, we give a composition theorem showing how to build a hash function family satisfying the two assumptions we need on G from two hash functions that individually satisfy the respective property. In particular, in light of the results of Chen and Mao [36], this reduces the task of constructing such a G to constructing an unpredictability-preserving hash function family that satisfies 1-DINJEXT2 for a derived FO wrapping triple.

Proof outline. The main difficulty is that 1-DINJEXT2 does not let us replace the decryption oracle in one step. With a coin-based extractor, one usually swaps the decryption oracle for the extractor and argues that the adversary's view does not change. A distributional simulator runs offline, and it only guarantees that *one fixed output predicate accepts* with about the same probability as in the ideal inversion game. We therefore need to identify each of the events that could make the games differ, and bound each one by a separate application of 1-DINJEXT2, with an output predicate tailored to that event. The three 1-DINJEXT2 $[\Gamma_{\text{SE}}]$ adversaries in the proof share one run stage, which runs the IND-CCA2 adversary, and differ only in their output predicates. In each application, we get a separate simulator.

In more detail, the proof uses three games. Game G_0 is the real IND-CCA2 game. Its decryption oracle inverts F using sk , performs the re-evaluation check, and decrypts the symmetric component under the key given by the output of G .

Game G_1 is the ideal inversion game for the first of the three adversaries above, which we call $\mathcal{B}_1$; its output predicate checks whether the guess bit of the assumed IND-CCA2 adversary is equal to the challenge bit. Here, decryption queries are answered using the ideal inverter instead of sk , the challenge ciphertext comes from the challenge oracle, and every later query other than the challenge whose first component equals that of the challenge is rejected and recorded. The games G_0 and G_1 are identical unless one of two bad events occurs in G_1 .

The first event is that the ideal inverter returns a preimage x on which inversion under sk fails. Note that such a preimage is chosen by the adversary rather than chosen at random, so the usual correctness condition on F does not suffice. We apply 1-DINJEXT2 with a predicate that tests for this event, and then bound the probability of it in the simulation game using adaptive correctness. The adaptive-correctness adversary receives sk , so it can run the simulation game itself and find a reported preimage on which inversion fails.

The second event is that one of the rejected queries would have been accepted in G_0 . Such a query has the form (c_1^*, c_2') with $c_2' \neq c_2^*$, where $C^* = (c_1^*, c_2^*)$ is the challenge ciphertext. Decryption in G_0 recovers the challenge preimage x^* from c_1^* and decrypts c_2' under the challenge key $K^* = G(K_G, x^*)$, so the event is that c_2' is a valid ciphertext under K^* . We again apply 1-DINJEXT2 with a predicate that tests for this event. In the simulation game, UCE security of G for the source S_F allows us to replace the challenge symmetric key with a random one, and then ciphertext integrity of SE bounds the probability of the event in question.

Game G_2 is the simulation game for $\mathcal{B}_1$. The simulator that 1-DINJEXT2 guarantees for $\mathcal{B}_1$ produces a transcript *without* an inversion oracle, and the same output predicate is checked on it. Since G_1 and G_2 are the two games of the 1-DINJEXT2 definition for the same adversary, the 1-DINJEXT2 advantage directly bounds the difference in the probabilities that they output 1. As Game G_2 has no decryption oracle, the simulator, with its challenge query forwarded to the IND-CPA challenge oracle, is an IND-CPA adversary against FO. Summing the bounds gives the theorem.

Proof of Theorem 6.2. Recall that $\text{Adv}_{\text{FO}}^{\text{ind-cca2}}(\mathcal{A}) = |2 \Pr[b' = b] - 1|$, where, as in the standard game, $\mathcal{A}$ submits a pair of equal-length messages in MsgSp_{SE} and outputs a bit b' . Throughout the proof we write $\Gamma \stackrel{\text{def}}{=} \Gamma_{\text{SE}}$ and

$$\text{Adv}_{\text{dext}}(\mathcal{B}, \mathcal{R}) \stackrel{\text{def}}{=} \text{Adv}_{G, \text{Wrap}_{\text{FO}}}^{1\text{-DINJEXT2}[\Gamma]}(\mathcal{B}, \mathcal{R}).$$

For a transcript v , let $\text{Guess}(v)$ be the first component b' if v parses as (b', C, τ, μ) with $b' \in \{0, 1\}$, and 0 otherwise. The output predicates of $\mathcal{B}_1^{\text{corr}}$ and $\mathcal{B}_1^{\text{maul}}$ below, and the distinguisher $\mathcal{D}$, return 0 on a transcript that does not parse in the stated form.

Game G_0 . This is the IND-CCA2 game for FO, shown in Figure 13. Its decryption oracle rejects the challenge ciphertext, inverts c_1 under sk , performs the re-evaluation check, and decrypts c_2 under the key $G(K_G, x)$.

<u>Game $G_0^{\mathcal{A}}$</u>	<u>Dec(c)</u>
$b \leftarrow_{\$} \{0, 1\}$	Parse c as (c_1, c_2)
$(pk, sk) \leftarrow_{\$} \text{F.Kg}(1^\lambda)$	If parsing fails: return $\perp$
$K_G \leftarrow_{\$} G.\text{Kg}(1^\lambda)$	If $C^* \neq \perp$ and $c = C^*$: return $\perp$
$\widehat{pk} \leftarrow (pk, K_G)$	$x \leftarrow \text{F.Inv}(sk, c_1)$
$C^* \leftarrow \perp$	If $x = \perp$: return $\perp$
$(st, m_0, m_1) \leftarrow_{\$} \mathcal{A}_1^{\text{Dec}}(1^\lambda, \widehat{pk})$	If $\text{F.Eval}(pk, x) \neq c_1$: return $\perp$
$x^* \leftarrow_{\$} \text{F.Dom}$	$K \leftarrow G(K_G, x)$
$c_1^* \leftarrow \text{F.Eval}(pk, x^*)$	Return $\text{SE.Dec}(K, c_2)$
$K^* \leftarrow G(K_G, x^*)$	
$c_2^* \leftarrow_{\$} \text{SE.Enc}(K^*, m_b)$	
$C^* \leftarrow (c_1^*, c_2^*)$	
$b' \leftarrow_{\$} \mathcal{A}_2^{\text{Dec}}(st, C^*)$	
Return $(b = b')$	

Figure 13: **Game G_0 :** the IND-CCA2 game for FO.

The 1-DINJEXT2 adversaries. The proof uses 1-DINJEXT2[Γ] for the hash function G and the FO wrapping triple Wrap_{FO} , in which the hash key is K_G , the setup parameter is the public key pk , and the hidden setup state is $aux = ((sk, x^*), b)$. On a challenge input (m_0, m_1) , the challenge oracle returns

$$C^* = (c_1^*, c_2^*) = (\text{F.Eval}(pk, x^*), \text{SE.Enc}(G(K_G, x^*), m_b; \rho^*))$$

for fresh coins ρ^* , which is distributed exactly as the challenge ciphertext in G_0 . After this query, the inversion oracle returns $\perp$ on C^* and on every other image whose preimage is x^* .

The three adversaries $\mathcal{B}_1$, $\mathcal{B}_1^{\text{corr}}$, and $\mathcal{B}_1^{\text{maul}}$ share the run stage $\mathcal{B}_{\text{run}}$ of Figure 14 and differ only in their output predicates. The run stage runs $\mathcal{A}$, obtains the challenge ciphertext from the challenge oracle, and answers the decryption queries of $\mathcal{A}$ with the procedure Dec' . This procedure rejects the challenge ciphertext, and for every other query made after the challenge whose first component is c_1^* , it records the second component in the set μ and returns $\perp$. Every remaining query that parses goes to the inversion oracle, and for each answer $x \neq \perp$, the procedure records the query and x in the set τ and returns $\text{SE.Dec}(G(K_G, x), c_2)$. The run stage outputs $v = (b', C^*, \tau, \mu)$.

The three output predicates, shown in Figure 15, test for the three events whose probabilities the proof bounds. The output predicate of $\mathcal{B}_1$ checks the guess of $\mathcal{A}$. The output predicate of $\mathcal{B}_1^{\text{corr}}$ checks whether some recorded preimage disagrees with inversion under sk . The output predicate of $\mathcal{B}_1^{\text{maul}}$ checks whether the reported ciphertext is $(c_1^*, \tilde{c}_2)$ with $\tilde{c}_2$ valid under the challenge key, and some recorded second component other than $\tilde{c}_2$ is also valid under that key. For the simulators $\mathcal{R}_1$, $\mathcal{R}_1^{\text{corr}}$, and $\mathcal{R}_1^{\text{maul}}$ of the theorem, we write Δ_1 , Δ_1^{corr} , and Δ_1^{maul} for $\text{Adv}_{\text{dext}}(\mathcal{B}_1, \mathcal{R}_1)$, $\text{Adv}_{\text{dext}}(\mathcal{B}_1^{\text{corr}}, \mathcal{R}_1^{\text{corr}})$, and $\text{Adv}_{\text{dext}}(\mathcal{B}_1^{\text{maul}}, \mathcal{R}_1^{\text{maul}})$.

$\mathcal{B}_{\text{run}}^{O_{\text{chal}}, O_{\text{inv}}}(1^\lambda, K_G, pk)$	$\text{Dec}'(c)$
$\widehat{pk} \leftarrow (pk, K_G); \tau \leftarrow \emptyset; \mu \leftarrow \emptyset; C^* \leftarrow \perp$	Parse c as (c_1, c_2)
$(st, m_0, m_1) \leftarrow \mathcal{A}_1^{\text{Dec}'}(1^\lambda, \widehat{pk})$	If parsing fails: return $\perp$
$C^* \leftarrow O_{\text{chal}}(m_0, m_1); \text{parse } C^* = (c_1^*, c_2^*)$	If $C^* \neq \perp$ and $c = C^*$: return $\perp$
$b' \leftarrow \mathcal{A}_2^{\text{Dec}'}(st, C^*)$	If $C^* \neq \perp$ and $c_1 = c_1^*$:
Return $v = (b', C^*, \tau, \mu)$	$\mu \leftarrow \mu \cup \{c_2\}$
	Return $\perp$
	$x \leftarrow O_{\text{inv}}(c)$
	If $x = \perp$: return $\perp$
	$\tau \leftarrow \tau \cup \{(c, x)\}$
	$K \leftarrow G(K_G, x)$
	Return $\text{SE.Dec}(K, c_2)$

Figure 14: **The shared run stage $\mathcal{B}_{\text{run}}$** of the three 1-DINJEXT2[Γ] adversaries for FO, and the procedure Dec' that answers the decryption queries of $\mathcal{A}$.

Game G_1 . This is the ideal inversion game $\text{Inv-1-DINJEXT2}[\Gamma]$ for $\mathcal{B}_1$. It outputs $[b = b']$, where b' is the guess of $\mathcal{A}$, whose decryption queries are now answered by Dec' . In G_1 , let Bad_{corr} be the event that some $((c_1, c_2), x) \in \tau$ has $x \in \text{F.Dom}$, $\text{F.Eval}(pk, x) = c_1$, and $\text{F.Inv}(sk, c_1) \neq x$. Let Bad_{maul} be the event that some $c'_2 \in \mu$ has $\text{SE.Dec}(K^*, c'_2) \neq \perp$, where $K^* = G(K_G, x^*)$. Every $c'_2 \in \mu$ differs from c_2^* , since a second component c_2 is recorded only for a query $(c_1^*, c_2) \neq C^*$.

$\mathcal{B}_{1,\text{out}}(K_G, pk, aux, v)$ Parse $aux = ((sk, x^*), b)$ Return $[b = \text{Guess}(v)]$ $\mathcal{B}_{1,\text{out}}^{\text{corr}}(K_G, pk, aux, v)$ Parse $aux = ((sk, x^*), b)$ and $v = (b', C, \tau, \mu)$ Return 1 iff some $((c_1, c_2), x) \in \tau$ has $x \in \text{F.Dom}, \text{F.Eval}(pk, x) = c_1,$ and $\text{F.Inv}(sk, c_1) \neq x$	$\mathcal{B}_{1,\text{out}}^{\text{maul}}(K_G, pk, aux, v)$ Parse $aux = ((sk, x^*), b)$ and $v = (b', C, \tau, \mu)$ $c_1^* \leftarrow \text{F.Eval}(pk, x^*); K^* \leftarrow G(K_G, x^*)$ Parse $C = (c_1, \tilde{c}_2)$; if $c_1 \neq c_1^*$: return 0 If $\text{SE.Dec}(K^*, \tilde{c}_2) = \perp$: return 0 Return 1 iff some $c'_2 \in \mu$ has $c'_2 \neq \tilde{c}_2$ and $\text{SE.Dec}(K^*, c'_2) \neq \perp$
--	--

Figure 15: The output predicates of $\mathcal{B}_1$, $\mathcal{B}_1^{\text{corr}}$, and $\mathcal{B}_1^{\text{maul}}$.

The ideal inversion games for $\mathcal{B}_1^{\text{corr}}$ and $\mathcal{B}_1^{\text{maul}}$ run exactly as G_1 and differ only in the predicate applied to v at the end. The predicate of $\mathcal{B}_1^{\text{corr}}$ is exactly the definition of Bad_{corr} . In the predicate of $\mathcal{B}_1^{\text{maul}}$, the reported ciphertext is C^* , so that $\tilde{c}_2 = c_2^*$, which is valid under K^* by correctness of SE, and the predicate therefore accepts exactly when Bad_{maul} occurs. Hence

$$\begin{aligned} \Pr[\text{Bad}_{\text{corr}} \text{ in } G_1] &= \Pr[\text{Inv-1-DINJEXT2}[\Gamma]^{\mathcal{B}_1^{\text{corr}}} \Rightarrow 1], \\ \Pr[\text{Bad}_{\text{maul}} \text{ in } G_1] &= \Pr[\text{Inv-1-DINJEXT2}[\Gamma]^{\mathcal{B}_1^{\text{maul}}} \Rightarrow 1]. \end{aligned} \tag{21}$$

Claim 6.3. $|\Pr[G_1 \Rightarrow 1] - \Pr[G_0 \Rightarrow 1]| \leq \Pr[\text{Bad}_{\text{corr}} \text{ in } G_1] + \Pr[\text{Bad}_{\text{maul}} \text{ in } G_1].$

Proof. Both games output $[b = b']$, and they sample the keys, the challenge bit, and the challenge ciphertext in the same way. We show that the two decryption oracles give the same answer to every query, unless the query itself causes Bad_{corr} or Bad_{maul} . The two games are therefore identical until the first such query, and the claim follows by the identical-until-bad argument. The argument concerns only the classical queries and answers, so it applies to a QPT adversary as well.

Consider a query $c = (c_1, c_2)$. If c does not parse, or if $c = C^*$ after the challenge, both games return $\perp$.

Suppose next that the challenge has not been made, or that $c_1 \neq c_1^*$. Then Dec' sends c to the inversion oracle. If the oracle returns $x \neq \perp$, then $C_{pk}(x, G(K_G, x), c) = 1$, so $x \in \text{F.Dom}$ and $\text{F.Eval}(pk, x) = c_1$, and Dec' records (c, x) in τ and returns $\text{SE.Dec}(G(K_G, x), c_2)$. Unless Bad_{corr} occurs, $\text{F.Inv}(sk, c_1) = x$, so G_0 passes the re-evaluation check and returns the same value. If the oracle returns $\perp$, then G_0 also returns $\perp$. Indeed, suppose G_0 returned a value other than $\perp$. Then $x = \text{F.Inv}(sk, c_1)$ satisfies $\text{F.Eval}(pk, x) = c_1$ and $\text{SE.Dec}(G(K_G, x), c_2) \neq \perp$. By image-closedness of SE, there are $m \in \text{MsgSp}_{\text{SE}}$ and $\rho \in \text{CoinSp}_{\text{SE}}$ with $c_2 = \text{SE.Enc}(G(K_G, x), m; \rho)$, so $c = g_{pk}^{\text{FO}}(x, G(K_G, x), m; \rho)$ and $C_{pk}(x, G(K_G, x), c) = 1$. The inversion oracle returns $\perp$ on such a query only if it is C^* or its preimage x is x^* . The first is excluded. The second is impossible, because before the challenge no preimage is registered, and after it $c_1 \neq c_1^* = \text{F.Eval}(pk, x^*)$ gives $x \neq x^*$. So the inversion oracle would have returned x , a contradiction.

Finally, suppose the challenge has been made and $c_1 = c_1^*$ with $c \neq C^*$, so $c_2 \neq c_2^*$. Then G_1 records c_2 in μ and returns $\perp$. In G_0 , let $x = \text{F.Inv}(sk, c_1^*)$. If $x = \perp$, then G_0 returns $\perp$. If $x \neq \perp$ and $x \neq x^*$, then $\text{F.Eval}(pk, x) \neq \text{F.Eval}(pk, x^*) = c_1^*$ because $\text{F.Eval}(pk, \cdot)$ is injective, so the re-evaluation check fails and G_0 returns $\perp$. If $x = x^*$, then G_0 returns $\text{SE.Dec}(K^*, c_2)$, which is $\perp$ unless this query causes Bad_{maul} . $\square$

Claim 6.4. For every PPT simulator $\mathcal{R}_1^{\text{corr}}$, there is a PPT adaptive-correctness adversary $\mathcal{A}^{\text{corr}}$ such that

$$\Pr[\text{Bad}_{\text{corr}} \text{ in } G_1] \leq \Delta_1^{\text{corr}} + \text{Adv}_{\mathbb{F}}^{\text{adapt-corr}}(\mathcal{A}^{\text{corr}}).$$

Proof. By Equation (21) and the definition of Δ_1^{corr} ,

$$\Pr[\text{Bad}_{\text{corr}} \text{ in } G_1] \leq \Pr[\text{Sim-1-DINJEXT2}[\Gamma]^{\mathcal{B}_1^{\text{corr}}, \mathcal{R}_1^{\text{corr}}} \Rightarrow 1] + \Delta_1^{\text{corr}}.$$

The adversary $\mathcal{A}^{\text{corr}}$, given (pk, sk) , samples $K_G \leftarrow \$ G.\text{Kg}(1^\lambda)$, $x^* \leftarrow \$ \mathbb{F}.\text{Dom}$, and $b \leftarrow \$ \{0, 1\}$. It runs $\mathcal{R}_1^{\text{corr}}(1^\lambda, K_G, pk)$ and answers its challenge query as the challenge oracle does, which it can because it knows K_G , x^* , and b . From the transcript it outputs the first x that appears in a pair $((c_1, c_2), x) \in \tau$ with $x \in \mathbb{F}.\text{Dom}$, $\mathbb{F}.\text{Eval}(pk, x) = c_1$, and $\mathbb{F}.\text{Inv}(sk, c_1) \neq x$, or an arbitrary element of $\mathbb{F}.\text{Dom}$ if there is none. The values it samples have the same distribution as in the simulation game. When the predicate accepts, the x it outputs satisfies $\mathbb{F}.\text{Inv}(sk, \mathbb{F}.\text{Eval}(pk, x)) \neq x$. So $\mathcal{A}^{\text{corr}}$ wins the adaptive-correctness game (Definition 2.8) whenever the predicate of $\mathcal{B}_1^{\text{corr}}$ accepts in the simulation game. It is PPT because $\mathcal{R}_1^{\text{corr}}$ is. $\square$

Claim 6.5. For every PPT simulator $\mathcal{R}_1^{\text{maul}}$, there are a PPT UCE adversary $\mathcal{B}_2 = (S_{\mathbb{F}}, \mathcal{D})$, with $S_{\mathbb{F}}$ the source of Section 6.1, and a PPT INT-CTXT adversary $\mathcal{B}_4$ such that

$$\Pr[\text{Bad}_{\text{maul}} \text{ in } G_1] \leq \Delta_1^{\text{maul}} + \text{Adv}_G^{\text{uce}}(\mathcal{B}_2) + \text{Adv}_{\text{SE}}^{\text{INT-CTXT}}(\mathcal{B}_4).$$

Proof. By Equation (21) and the definition of Δ_1^{maul} ,

$$\Pr[\text{Bad}_{\text{maul}} \text{ in } G_1] \leq p_{\text{sim}} + \Delta_1^{\text{maul}}, \quad \text{where } p_{\text{sim}} = \Pr[\text{Sim-1-DINJEXT2}[\Gamma]^{\mathcal{B}_1^{\text{maul}}, \mathcal{R}_1^{\text{maul}}} \Rightarrow 1].$$

We bound p_{sim} in two steps.

First, the distinguisher $\mathcal{D}$ of Figure 16 receives the leakage (pk, c_1^*, K^*) of $S_{\mathbb{F}}$ and the hash key K_G . It samples the challenge bit, runs $\mathcal{R}_1^{\text{maul}}(1^\lambda, K_G, pk)$, answers its challenge query with $(c_1^*, \text{SE}.\text{Enc}(K^*, m_b))$, and applies the output predicate of $\mathcal{B}_1^{\text{maul}}$ to the transcript, using K^* . In the UCE game with challenge bit $\beta = 1$, we have $K^* = G(K_G, x^*)$, so $\mathcal{D}$ runs the simulation game for $\mathcal{B}_1^{\text{maul}}$ exactly. With $\beta = 0$, the key K^* is uniform in $\{0, 1\}^{\ell_K}$ and independent of (pk, x^*, K_G) . Writing p_β for the probability that $\mathcal{D}$ outputs 1 when the UCE challenge bit is β , the definition of the UCE advantage gives

$$p_{\text{sim}} = p_1 \leq p_0 + \text{Adv}_G^{\text{uce}}(\mathcal{B}_2).$$

Second, the adversary $\mathcal{B}_4$ of Figure 17 runs the $\beta = 0$ execution of $\mathcal{D}$ with the key of the INT-CTXT game in place of K^* . That key is uniform in $\{0, 1\}^{\ell_K}$ by Definition 2.2, so the execution is distributed as in the case $\beta = 0$. Adversary $\mathcal{B}_4$ samples pk, x^*, K_G , and the challenge bit itself, answers the challenge query of the simulator with (c_1^*, d) for $d = \text{Enc}(m_b)$, and after the simulator stops, submits verification queries. Suppose that $\mathcal{D}$ would output 1. Then the reported ciphertext is $(c_1^*, \tilde{c}_2)$ with $\tilde{c}_2$ valid under K^* , and some $c'_2 \in \mu$ with $c'_2 \neq \tilde{c}_2$ is valid under K^* . If no challenge query was answered, so that $d = \perp$, or if $\tilde{c}_2 \neq d$, then $\tilde{c}_2$ is a valid ciphertext that Enc did not return. If $\tilde{c}_2 = d$, then $c'_2 \neq d$ is a valid ciphertext that Enc did not return. In either case, $\mathcal{B}_4$ submits it to Vfy and wins, so $p_0 \leq \text{Adv}_{\text{SE}}^{\text{INT-CTXT}}(\mathcal{B}_4)$. Both $\mathcal{D}$ and $\mathcal{B}_4$ are PPT because $\mathcal{R}_1^{\text{maul}}$ is. Combining the three inequalities proves the claim. $\square$

<u>Source $S_F^{\text{HASH}}(1^\lambda)$</u> $(pk, sk) \leftarrow \$ \text{F.Kg}(1^\lambda)$ $x^* \leftarrow \$ \text{F.Dom}; c_1^* \leftarrow \text{F.Eval}(pk, x^*)$ $K^* \leftarrow \text{HASH}(x^*)$ Return $L = (pk, c_1^*, K^*)$	<u>Distinguisher $\mathcal{D}(L, K_G)$</u> $(pk, c_1^*, K^*) \leftarrow L; b \leftarrow \$ \{0, 1\}; q_{\text{chal}} \leftarrow 0$ $v \leftarrow \$ (\mathcal{R}_1^{\text{maul}})^{O_{\text{chal}}}(1^\lambda, K_G, pk)$ Parse $v = (b', C, \tau, \mu)$ and $C = (c_1, \tilde{c}_2)$ If $c_1 \neq c_1^*$ or $\text{SE.Dec}(K^*, \tilde{c}_2) = \perp$: return 0 Return 1 iff some $c'_2 \in \mu$ has $c'_2 \neq \tilde{c}_2$ and $\text{SE.Dec}(K^*, c'_2) \neq \perp$ <u>$O_{\text{chal}}(m_0, m_1)$</u> If $(m_0, m_1) \notin \mathcal{W}'$: return $\perp$ $q_{\text{chal}} \leftarrow q_{\text{chal}} + 1$ If $q_{\text{chal}} > 1$: return $\perp$ Return $(c_1^*, \text{SE.Enc}(K^*, m_b))$
---	---

Figure 16: UCE adversary $\mathcal{B}_2 = (S_F, \mathcal{D})$ for Claim 6.5.

<u>$\mathcal{B}_4^{\text{Enc, Vfy}}(1^\lambda)$</u> $(pk, sk) \leftarrow \$ \text{F.Kg}(1^\lambda); K_G \leftarrow \$ G.\text{Kg}(1^\lambda)$ $x^* \leftarrow \$ \text{F.Dom}; c_1^* \leftarrow \text{F.Eval}(pk, x^*)$ $b \leftarrow \$ \{0, 1\}; d \leftarrow \perp; q_{\text{chal}} \leftarrow 0$ $v \leftarrow \$ (\mathcal{R}_1^{\text{maul}})^{O_{\text{chal}}}(1^\lambda, K_G, pk)$ Parse $v = (b', C, \tau, \mu)$ and $C = (c_1, \tilde{c}_2)$ If $c_1 \neq c_1^*$: return If $\tilde{c}_2 \neq d$: $\text{Vfy}(\tilde{c}_2)$ For each $c'_2 \in \mu$ with $c'_2 \neq d$: $\text{Vfy}(c'_2)$	<u>$O_{\text{chal}}(m_0, m_1)$</u> If $(m_0, m_1) \notin \mathcal{W}'$: return $\perp$ $q_{\text{chal}} \leftarrow q_{\text{chal}} + 1$ If $q_{\text{chal}} > 1$: return $\perp$ $d \leftarrow \text{Enc}(m_b)$ Return (c_1^*, d)
--	---

Figure 17: INT-CTXT adversary $\mathcal{B}_4$ for Claim 6.5.

Game G_2 . This is the simulation game $\text{Sim-1-DINJEXT2}[\Gamma]$ for $\mathcal{B}_1$ and $\mathcal{R}_1$. The simulator is given $(1^\lambda, K_G, pk)$, has access to the challenge oracle, and outputs a transcript v on its own, and the game outputs $[b = \text{Guess}(v)]$. Since G_1 and G_2 are the two games of the 1-DINJEXT2 definition for $\mathcal{B}_1$,

$$|\Pr[G_2 \Rightarrow 1] - \Pr[G_1 \Rightarrow 1]| \leq \Delta_1.$$

Since G_2 has no decryption oracle, it can be run by the IND-CPA adversary $\mathcal{B}_3$ of Figure 18, which receives the public key (pk, K_G) of FO, runs $\mathcal{R}_1(1^\lambda, K_G, pk)$, answers its challenge query with its own left-or-right oracle, and outputs $\text{Guess}(v)$. The challenge oracle accepts only pairs of equal-length messages, so every query that $\mathcal{B}_3$ forwards is a legal IND-CPA query. The reply is an encryption of m_b under a fresh uniform preimage, which is distributed as the challenge ciphertext in G_2 , since the simulator sees x^* only through that ciphertext. Hence

$$|2\Pr[G_2 \Rightarrow 1] - 1| = \text{Adv}_{\text{FO}}^{\text{ind-cpa}}(\mathcal{B}_3).$$

Combining the bounds. By the triangle inequality,

$$\begin{aligned} \text{Adv}_{\text{FO}}^{\text{ind-cca2}}(\mathcal{A}) &= |2\Pr[G_0 \Rightarrow 1] - 1| \leq 2|\Pr[G_0 \Rightarrow 1] - \Pr[G_1 \Rightarrow 1]| + 2|\Pr[G_1 \Rightarrow 1] - \Pr[G_2 \Rightarrow 1]| \\ &\quad + |2\Pr[G_2 \Rightarrow 1] - 1|. \end{aligned}$$

$\mathcal{B}_3^{\text{LR}(\cdot, \cdot)}(1^\lambda, \widehat{pk})$ Parse $\widehat{pk} = (pk, K_G)$; $q_{\text{chal}} \leftarrow 0$ $v \leftarrow \mathcal{R}_1^{O_{\text{chal}}}(1^\lambda, K_G, pk)$ Return $\text{Guess}(v)$	$O_{\text{chal}}(m_0, m_1)$ If $(m_0, m_1) \notin \mathcal{W}'$: return $\perp$ $q_{\text{chal}} \leftarrow q_{\text{chal}} + 1$ If $q_{\text{chal}} > 1$: return $\perp$ Return $\text{LR}(m_0, m_1)$
---	--

Figure 18: IND-CPA adversary $\mathcal{B}_3$ for the final step.

By Claims 6.3 to 6.5 and the two bounds for G_2 ,

$$\begin{aligned} \text{Adv}_{\text{FO}}^{\text{ind-cca}2}(\mathcal{A}) &\leq 2(\Delta_1 + \Delta_1^{\text{corr}} + \Delta_1^{\text{maul}}) + 2\text{Adv}_{\text{F}}^{\text{adapt-corr}}(\mathcal{A}^{\text{corr}}) \\ &\quad + 2\text{Adv}_{\text{G}}^{\text{uce}}(\mathcal{B}_2) + \text{Adv}_{\text{FO}}^{\text{ind-cpa}}(\mathcal{B}_3) + 2\text{Adv}_{\text{SE}}^{\text{INT-CTXT}}(\mathcal{B}_4). \end{aligned}$$

The adversary $\mathcal{A}^{\text{corr}}$ is PPT, so δ -adaptive correctness gives $\text{Adv}_{\text{F}}^{\text{adapt-corr}}(\mathcal{A}^{\text{corr}}) \leq \delta$, which proves the claimed bound. $\square$

6.3 IND-CPA Security of FO Hybrid Encryption from UCE

Theorem 6.2 assumes IND-CPA security of $\text{FO}[\text{F}, \text{SE}, \text{G}]$. We show that it follows from one-wayness of F , IND-CPA security of SE , and $\mathcal{S}_1^{\text{scup}}$ -UCE security of G , which implies the UCE assumption of the theorem when F is one-way.

Proposition 6.6 (IND-CPA security of $\text{FO}[\text{F}, \text{SE}, \text{G}]$ from UCE). *Let F be a one-way injective trapdoor function. Let SE be an IND-CPA-secure symmetric encryption scheme and G an $\mathcal{S}_1^{\text{scup}}$ -UCE-secure keyed hash function family. Then for every IND-CPA adversary $\mathcal{A}$ there exist a single-query UCE adversary $\mathcal{B}_1 = (\mathcal{S}, \mathcal{D})$ with $\mathcal{S} \in \mathcal{S}_1^{\text{scup}}$ and an IND-CPA adversary $\mathcal{B}_2$ against SE such that*

$$\text{Adv}_{\text{FO}}^{\text{ind-cpa}}(\mathcal{A}) \leq 2\text{Adv}_{\text{G}}^{\text{uce}}(\mathcal{B}_1) + \text{Adv}_{\text{SE}}^{\text{ind-cpa}}(\mathcal{B}_2).$$

The same bound holds for QPT adversaries under post-quantum one-wayness of F , post-quantum $\mathcal{S}_1^{\text{scup}}$ -UCE security of G , and post-quantum IND-CPA security of SE .

Proof. We bound $\text{Adv}_{\text{FO}}^{\text{ind-cpa}}(\mathcal{A}) = |2\Pr[b' = b] - 1|$ via a sequence of two games.

Games G_0 (Real IND-CPA) and G_1 (Random challenge key). G_0 is the standard IND-CPA game for FO , shown in Figure 19 (left): the adversary receives the public key $\widehat{pk} = (pk, K_G)$ and a challenge ciphertext (c_1^*, c_2^*) computed using the real hash function G . G_1 replaces the challenge key $K^* = \text{G}(K_G, x^*)$ with a uniformly random key $\tilde{K} \leftarrow \text{SE.Kg}(1^\lambda)$; the only change from G_0 is boxed in Figure 19 (right).

Claim 6.7. There exists a UCE adversary $\mathcal{B}_1 = (\mathcal{S}_{\text{F}}, \mathcal{D})$ such that

$$|\Pr[G_1 \Rightarrow 1] - \Pr[G_0 \Rightarrow 1]| \leq \text{Adv}_{\text{G}}^{\text{uce}}(\mathcal{B}_1),$$

where $\mathcal{S}_{\text{F}}$ is the single-query source of Section 6.1, which belongs to $\mathcal{S}_1^{\text{scup}}$.

Proof of Claim 6.7. The source $\mathcal{S}_{\text{F}}$ and distinguisher $\mathcal{D}$ are defined in Figure 20.

When $\text{HASH} = \text{G}(K_G, \cdot)$, $K^* = \text{G}(K_G, x^*)$ and $\mathcal{D}$ perfectly simulates G_0 . When HASH is a random function, K^* is uniform and $\mathcal{D}$ simulates G_1 .

Game $G_0^{\mathcal{A}}$	Game $G_1^{\mathcal{A}}$
$b \leftarrow \{0, 1\}$	$b \leftarrow \{0, 1\}$
$(pk, sk) \leftarrow \text{F.Kg}(1^\lambda)$	$(pk, sk) \leftarrow \text{F.Kg}(1^\lambda)$
$K_G \leftarrow G.\text{Kg}(1^\lambda)$	$K_G \leftarrow G.\text{Kg}(1^\lambda)$
$\widehat{pk} \leftarrow (pk, K_G)$	$\widehat{pk} \leftarrow (pk, K_G)$
$(st, m_0, m_1) \leftarrow \mathcal{A}_1(1^\lambda, \widehat{pk})$	$(st, m_0, m_1) \leftarrow \mathcal{A}_1(1^\lambda, \widehat{pk})$
$x^* \leftarrow \text{F.Dom}$	$x^* \leftarrow \text{F.Dom}$
$c_1^* \leftarrow \text{F.Eval}(pk, x^*)$	$c_1^* \leftarrow \text{F.Eval}(pk, x^*)$
$K^* \leftarrow G(K_G, x^*)$	$\tilde{K} \leftarrow \text{SE.Kg}(1^\lambda)$
$c_2^* \leftarrow \text{SE.Enc}(K^*, m_b)$	$c_2^* \leftarrow \text{SE.Enc}(\tilde{K}, m_b)$
$C^* \leftarrow (c_1^*, c_2^*)$	$C^* \leftarrow (c_1^*, c_2^*)$
$b' \leftarrow \mathcal{A}_2(st, C^*)$	$b' \leftarrow \mathcal{A}_2(st, C^*)$
Return $(b = b')$	Return $(b = b')$

Figure 19: **Left:** Game G_0 , the real IND-CPA game for FO. **Right:** Game G_1 , with a uniform challenge key.

Source $S_F^{\text{HASH}}(1^\lambda)$	Distinguisher $\mathcal{D}(L, K_G)$
$(pk, sk) \leftarrow \text{F.Kg}(1^\lambda)$	$(pk, c_1^*, K^*) \leftarrow L$
$x^* \leftarrow \text{F.Dom}$	$\widehat{pk} \leftarrow (pk, K_G)$
$c_1^* \leftarrow \text{F.Eval}(pk, x^*)$	$b \leftarrow \{0, 1\}$
$K^* \leftarrow \text{HASH}(x^*)$	$(st, m_0, m_1) \leftarrow \mathcal{A}_1(1^\lambda, \widehat{pk})$
Return (pk, c_1^*, K^*)	$c_2^* \leftarrow \text{SE.Enc}(K^*, m_b)$
	$C^* \leftarrow (c_1^*, c_2^*)$
	$b' \leftarrow \mathcal{A}_2(st, C^*)$
	Return $(b = b')$

Figure 20: UCE adversary $\mathcal{B}_1 = (S_F, \mathcal{D})$ for the $G_0 \rightarrow G_1$ transition.

S_F is **strongly computationally unpredictable**. We show $S_F \in \mathcal{S}_1^{\text{scup}}$ in the sense of Figure 5. S_F 's single oracle query is x^* , and its leakage is $L = (pk, c_1^*, K^*)$ with $K^* = \text{HASH}(x^*)$. In the prediction game HASH is a uniform random function, so K^* is uniform in the hash-output space and independent of (pk, x^*, c_1^*) . The strong prediction view adds only the single oracle response, and that response is already included in $L = (pk, c_1^*, K^*)$. A predictor $\mathcal{P}$ winning $\text{Pred}_{S_F, \mathcal{P}}^{1\text{-scup}}$ therefore outputs x^* from (pk, c_1^*, K^*) , so one can invert F as follows: given $(pk, c_1^* = \text{F.Eval}(pk, x^*))$, sample $\tilde{K}$ uniformly in the hash-output space, run $\mathcal{P}(pk, c_1^*, \tilde{K})$, and return its output. Since F is one-way, $\Pr[\text{Pred}_{S_F, \mathcal{P}}^{1\text{-scup}} \Rightarrow 1]$ is negligible, so $S_F \in \mathcal{S}_1^{\text{scup}}$. $\square$

Completing the proof. In G_1 , the challenge ciphertext is $(c_1^*, \text{SE.Enc}(\tilde{K}, m_b))$ where $\tilde{K}$ is a uniform random key, independent of $\mathcal{A}$'s view. The adversary's advantage in G_1 is therefore bounded by the IND-CPA security of SE . The reduction $\mathcal{B}_2$ is shown in Figure 21. $\mathcal{B}_2$ perfectly simulates G_1 , so

$$|2 \Pr[G_1 \Rightarrow 1] - 1| \leq \text{Adv}_{\text{SE}}^{\text{ind-cpa}}(\mathcal{B}_2).$$

$\mathcal{B}_2^{\text{LR}(\cdot, \cdot)}(1^\lambda)$ $(pk, sk) \leftarrow \$ \text{F.Kg}(1^\lambda)$ $K_G \leftarrow \$ G.\text{Kg}(1^\lambda)$ $\widehat{pk} \leftarrow (pk, K_G)$ $(st, m_0, m_1) \leftarrow \$ \mathcal{A}_1(1^\lambda, \widehat{pk})$ $x^* \leftarrow \$ \text{F.Dom}$ $c_1^* \leftarrow \text{F.Eval}(pk, x^*)$ $c_2^* \leftarrow \text{LR}(m_0, m_1)$ $C^* \leftarrow (c_1^*, c_2^*)$ $b' \leftarrow \$ \mathcal{A}_2(st, C^*)$ Return b'

Figure 21: IND-CPA adversary $\mathcal{B}_2$ for the reduction from G_1 to IND-CPA security of SE.

Combining the two transitions via the triangle inequality,

$$\begin{aligned}
\text{Adv}_{\text{FO}}^{\text{ind-cpa}}(\mathcal{A}) &= |2 \Pr[G_0 \Rightarrow 1] - 1| \\
&\leq 2 |\Pr[G_0 \Rightarrow 1] - \Pr[G_1 \Rightarrow 1]| + |2 \Pr[G_1 \Rightarrow 1] - 1| \\
&\leq 2 \text{Adv}_G^{\text{uce}}(\mathcal{B}_1) + \text{Adv}_{\text{SE}}^{\text{ind-cpa}}(\mathcal{B}_2).
\end{aligned}$$

This proves the proposition. $\square$

7 Distributional Injective Extractability in the ROM and QROM

We show that, under the conditions stated below, a random oracle satisfies distributional injective extractability for *every* Γ -PI wrapping triple whose honest algorithms have no oracle access. We also prove the corresponding result for the FO wrapping triple under a QROM version of the definition, in which the simulator answers some of the random-oracle queries. Both results hold even relative to an *oblivious sampling* oracle, which gives the adversary valid images whose preimages it does not know. This contrasts with our standard-model impossibility of post-quantum coin-based extractability, which uses oblivious sampling (Theorem 5.4). The ROM result also contrasts with the standard model, where no hash function family satisfies (distributional) injective extractability for every Γ -PI wrapping triple (Section 5.5). This can be viewed as analogous to UCE, which a random oracle satisfies for all computationally unpredictable sources [10], whereas no standard-model hash function family does, assuming indistinguishability obfuscation [24].

7.1 Modeling in the ROM and QROM

Throughout this section, $n = n(\lambda)$ is a fixed polynomial input length satisfying $n = \omega(\log \lambda)$, and the random oracle is an unkeyed function

$$G \leftarrow \$ \text{Funcs}(\{0, 1\}^n, \{0, 1\}^\ell).$$

The preimage domain $\mathcal{X}$ is a subset of $\{0, 1\}^n$.

Wrapping triples. A wrapping triple $(\mathcal{G}, \text{Param}, P)$ is as in Section 5. For the oblivious sampling oracle, we require that $\mathcal{X}$ and $\mathcal{R}$ can be randomly sampled in polynomial time. We also assume that the bound Γ of a Γ -PI wrapping function is nondecreasing in its length argument t . This loses no generality, because a Γ -PI wrapping function is also Γ' -PI for $\Gamma'(\lambda, t) = \max_{t' \leq t} \Gamma(\lambda, t')$, and it holds for Γ_{SE} (Section 2.4).

Oracle access. The setup sampler, preimage creator, wrapping function, and consistency predicate have no oracle access. The adversary has oracle access to G . In the ROM, so does the simulator; its QROM interface is specified below. The challenge oracle evaluates G at the preimage returned by the preimage creator and passes the result to the wrapping function, as in Figure 12. The oblivious sampling oracle defined below also evaluates G at the preimage it samples. These oracle calls do not give the wrapping function oracle access.

Games in the ROM. We adapt the games of Figure 12 with $G \leftarrow \text{Funcs}(\{0, 1\}^n, \{0, 1\}^\ell)$ in place of hash-key generation and with $G(x)$ in place of $\text{Hash.Eval}(\text{hk}, x)$. In the ROM and QROM, there is no hash key. The run stage and simulator receive $(1^\lambda, p)$, the output predicate receives $(1^\lambda, p, \text{aux}, v)$, and the preimage creator receives $(1^\lambda, p, \text{aux}, w')$. Oracle access is as stated above.

The FO wrapping triple. The FO wrapping triple Wrap_{FO} of Section 6.1 is built from a trapdoor function F and a symmetric encryption scheme SE whose algorithms have no oracle access. The random oracle G takes the place of the hash function G of FO. The hash function output length ℓ is the key length ℓ_K of SE , the preimage domain is $\mathcal{X} = F.\text{Dom} = \{0, 1\}^n$, the auxiliary-input space is $\mathcal{W} = \text{MsgSp}_{\text{SE}}$, and the coin space is $\mathcal{R} = \text{CoinSp}_{\text{SE}}$. The condition $n = \omega(\log \lambda)$ is necessary for F to be one-way, since guessing a uniform preimage succeeds with probability at least 2^{-n} . Throughout this section we assume that F is injective and SE is image-closed. Then Wrap_{FO} is Γ_{SE} -PI by Proposition 6.1, whose proof does not use one-wayness. The setup sampler Param_{FO} samples the challenge preimage x^* uniformly from $F.\text{Dom}$, independently of pk , and stores it in the hidden state $\text{aux} = ((sk, x^*), b)$. The preimage creator P_{FO} outputs this x^* .

Oblivious sampling. Our results hold even when the adversary has an oracle that returns valid images whose preimages it does not know. This oracle, $\mathcal{O}_{\text{samp}}$, takes an auxiliary input w , samples a fresh preimage x and fresh coins ρ , and returns the wrapped image of x and w (Figure 22). For the FO wrapping triple, w is a message m , and the oracle returns an FO ciphertext of m whose randomness x is hidden from the adversary.

Definition 7.1 (1-DINJEXT2 relative to $\mathcal{O}_{\text{samp}}$ in the ROM). *Let $(\mathcal{G}, \text{Param}, P)$ be a wrapping triple with $\mathcal{G}$ Γ -PI. All oracle queries are classical. For the games of Figure 22, the advantage of an adversary $\mathcal{B}$ with simulator $\mathcal{R}$ is*

$$\text{Adv}_{\text{RO}, \mathcal{G}, \text{Param}, P}^{\text{1-DINJEXT2-samp}}(\mathcal{B}, \mathcal{R}) \stackrel{\text{def}}{=} \left| \Pr[\text{Inv-1-DINJEXT2-samp}_{\text{RO}, \mathcal{G}}^{\mathcal{B}} \Rightarrow 1] - \Pr[\text{Sim-1-DINJEXT2}_{\text{RO}, \mathcal{G}}^{\mathcal{B}, \mathcal{R}} \Rightarrow 1] \right|.$$

We say that 1-DINJEXT2 $[\Gamma]$ holds relative to $\mathcal{O}_{\text{samp}}$ in the ROM for $(\mathcal{G}, \text{Param}, P)$ if for every PPT adversary $\mathcal{B}$ there is a PPT simulator $\mathcal{R}$ such that this advantage is negligible and $\mathcal{R}$ meets the condition on challenge queries of Definition 5.5 for $\mathcal{B}$. We write $\text{Adv}_{\text{RO}, \text{Wrap}_{\text{FO}}}^{\text{1-DINJEXT2-samp}}$ for the FO wrapping triple.

Game Inv-1-DINJEXT2-samp $_{\text{RO}, \mathcal{G}}^{\mathcal{B}}$	Game Sim-1-DINJEXT2 $_{\text{RO}, \mathcal{G}}^{\mathcal{B}, \mathcal{R}}$
$G \leftarrow \$ \text{Funcs}(\{0, 1\}^n, \{0, 1\}^\ell)$ $(p, aux) \leftarrow \$ \text{Param}(1^\lambda)$ $L_{\text{chal}} \leftarrow []; I_{\text{chal}} \leftarrow \emptyset$ $q_{\text{chal}} \leftarrow 0$ $v \leftarrow \$ \mathcal{B}_{\text{run}}^{G, O_{\text{chal}}, O_{\text{inv}}, O_{\text{samp}}}(1^\lambda, p)$ Return $\mathcal{B}_{\text{out}}^G(1^\lambda, p, aux, v)$	$G \leftarrow \$ \text{Funcs}(\{0, 1\}^n, \{0, 1\}^\ell)$ $(p, aux) \leftarrow \$ \text{Param}(1^\lambda)$ $L_{\text{chal}} \leftarrow []; I_{\text{chal}} \leftarrow \emptyset$ $q_{\text{chal}} \leftarrow 0$ $v \leftarrow \$ \mathcal{R}^{G, O_{\text{chal}}}(1^\lambda, p)$ Return $\mathcal{B}_{\text{out}}^G(1^\lambda, p, aux, v)$
$O_{\text{samp}}(w)$ If $w \notin \mathcal{W}$: return $\perp$ $x \leftarrow \$ X; \rho \leftarrow \$ \mathcal{R}$ Return $g_p(x, G(x), w; \rho)$	$O_{\text{chal}}(w')$ If $w' \notin \mathcal{W}'$: return $\perp$ $q_{\text{chal}} \leftarrow q_{\text{chal}} + 1$ If $q_{\text{chal}} > 1$: return $\perp$ $(x, w, \rho) \leftarrow \$ P(1^\lambda, p, aux, w')$ $z \leftarrow g_p(x, G(x), w; \rho)$ $I_{\text{chal}} \leftarrow I_{\text{chal}} \cup \{(x, z)\}$ Append z to L_{chal} ; return z
$O_{\text{inv}}(z)$ If $z \in L_{\text{chal}}$: return $\perp$ If $\nexists x \in X$ with $C_p(x, G(x), z) = 1$: return $\perp$ Let x be the unique x -preimage of z If $\exists (x^*, z^*) \in I_{\text{chal}}$ with $x = x^*$ and $z \neq z^*$: return $\perp$ Return x	

Figure 22: The ROM games for 1-DINJEXT2 relative to oblivious sampling, for a wrapping triple $(\mathcal{G}, \text{Param}, P)$. The challenge oracle is shared by the two games.

The QROM. We define the QROM version of the notion directly for the FO wrapping triple, because the definition uses the fact that its challenge preimage x^* is fixed at setup. The ideal inversion game is the one of the ROM for Wrap_{FO} , with quantum oracle access.

A simulator $\mathcal{R} = (\mathcal{R}.\text{Init}, \mathcal{R}.\text{RO}, \mathcal{R}.\text{Sim})$ consists of three QPT subroutines sharing a private quantum register S . The subroutine $\mathcal{R}.\text{Init}(1^\lambda)$ prepares S once, without oracle access.

The subroutine $\mathcal{R}.\text{RO}$ receives an n -qubit query register Q and an ℓ -qubit answer register Y , both disjoint from S . It applies a unitary of the form

$$U^{\mathcal{R}} = \sum_{x \in \{0,1\}^n} |x\rangle\langle x|_Q \otimes U_x,$$

where each U_x acts on Y and S . Thus it leaves the query input unchanged. Its circuit is part of $\mathcal{R}$ and makes no oracle calls. It returns Q, Y to the caller and retains S .

The subroutine $\mathcal{R}.\text{Sim}$ receives $(1^\lambda, p)$, may operate on S and its own workspace, and produces v . It has quantum access to the oracle $\tilde{G}$ defined below and classical access to O_{chal} . After it finishes, the game retains S to answer the queries of $\mathcal{B}_{\text{out}}$. The output predicate has no direct access to S .

The simulation game samples $y^* \leftarrow \$ \{0, 1\}^\ell$ independently of setup. It gives neither x^* nor y^* as an input to $\mathcal{R}$. Write $X_{y^*} |a\rangle = |a \oplus y^*\rangle$. The oracle $\tilde{G}$ applies the unitary

$$\tilde{U} = |x^*\rangle\langle x^*|_Q \otimes (X_{y^*} \otimes I_S) + \sum_{x \neq x^*} |x\rangle\langle x|_Q \otimes U_x \quad (22)$$

to Q, Y, S . This answers the branch $x = x^*$ using y^* and applies the simulator's oracle subroutine on all other branches, without measuring Q . The operator is unitary because its blocks on the orthogonal basis states of Q are unitary. The game implements it by coherently computing the

test $x = x^*$, applying the appropriate controlled circuit, and uncomputing the test. All calls use the same register S , without resetting it. Thus $\tilde{G}$ denotes a stateful quantum oracle; we do not assume that it evaluates a fixed function. In the simulation game, O_{chal} is the challenge oracle of Figure 22, except that it uses y^* in place of $G(x^*)$. This evaluation leaves S unchanged. The simulation game is given in Figure 23.

Game Inv-1-DINJEXT2-samp $_{\text{QRO}, \text{Wrap}_{\text{FO}}}^{\mathcal{B}}$	Game Sim-1-DINJEXT2 $_{\text{QRO}, \text{Wrap}_{\text{FO}}}^{\mathcal{B}, \mathcal{R}}$
$G \leftarrow \text{Funcs}(\{0, 1\}^n, \{0, 1\}^\ell)$ $(p, \text{aux}) \leftarrow \text{Param}(1^\lambda)$ $L_{\text{chal}} \leftarrow []; I_{\text{chal}} \leftarrow \emptyset$ $q_{\text{chal}} \leftarrow 0$ $v \leftarrow \mathcal{B}_{\text{run}}^{G, O_{\text{chal}}, O_{\text{inv}}, O_{\text{samp}}}(1^\lambda, p)$ Return $\mathcal{B}_{\text{out}}^G(1^\lambda, p, \text{aux}, v)$	$y^* \leftarrow \{0, 1\}^\ell$ $S \leftarrow \mathcal{R}.\text{Init}(1^\lambda)$ $(p, \text{aux}) \leftarrow \text{Param}(1^\lambda)$ $((sk, x^*), b) \leftarrow \text{aux}$ $L_{\text{chal}} \leftarrow []; I_{\text{chal}} \leftarrow \emptyset; q_{\text{chal}} \leftarrow 0$ $v \leftarrow \mathcal{R}.\text{Sim}^{\tilde{G}, O_{\text{chal}}}(1^\lambda, p)$ Return $\mathcal{B}_{\text{out}}^{\tilde{G}}(1^\lambda, p, \text{aux}, v)$ $\tilde{G}(Q, Y)$ Apply $\tilde{U}$ of Equation (22) to Q, Y, S Return Q, Y

Figure 23: The QROM games for 1-DINJEXT2 relative to oblivious sampling. The value y^* is sampled once and used for every query at x^* . Queries to G and $\tilde{G}$ may be quantum, and the routing for $\tilde{G}$ is applied coherently to each query input x ; queries to other oracles are classical.

Definition 7.2 (1-DINJEXT2 relative to O_{samp} in the QROM). *Let $(\mathcal{G}, \text{Param}, P) = \text{Wrap}_{\text{FO}}$. For the games of Figure 23, the advantage of an adversary $\mathcal{B}$ with simulator $\mathcal{R}$ is*

$$\text{Adv}_{\text{QRO}, \text{Wrap}_{\text{FO}}}^{1\text{-DINJEXT2-samp}}(\mathcal{B}, \mathcal{R}) \stackrel{\text{def}}{=} \left| \Pr[\text{Inv-1-DINJEXT2-samp}_{\text{QRO}, \text{Wrap}_{\text{FO}}}^{\mathcal{B}} \Rightarrow 1] - \Pr[\text{Sim-1-DINJEXT2}_{\text{QRO}, \text{Wrap}_{\text{FO}}}^{\mathcal{B}, \mathcal{R}} \Rightarrow 1] \right|.$$

We say that 1-DINJEXT2 $[\Gamma_{\text{SE}}]$ holds relative to O_{samp} in the QROM if for every QPT adversary $\mathcal{B}$ there is a QPT simulator $\mathcal{R}$ with the interface specified above such that this advantage is negligible and $\mathcal{R}$ meets the condition on challenge queries of Definition 5.5 for $\mathcal{B}$.

Discussion. In the ROM, the simulator runs the adversary with oracle access to G . It can therefore see the adversary's queries to G , which is the usual way of extracting in the ROM [15], but it does not see the queries inside the challenge oracle. So, as in the standard model, the challenge preimage goes to neither the adversary nor the simulator.

In the QROM, the simulator also answers the queries to G , because quantum queries cannot be recorded without disturbing the adversary's state. By answering the queries itself, the simulator can extract by measuring its own oracle state, following the online extraction of Don, Fehr, Majenz, and Schaffner (DFMS) [41]. It answers only the queries away from x^* . The game answers the query at x^* itself, since forwarding the challenge oracle's query to the simulator would reveal x^* , which the ROM simulator never sees. For the same reason, $\mathcal{R}.\text{RO}$ must not change the query input, so that the game can answer the point x^* on the corresponding branch of a query in superposition. The output predicate queries the same oracle $\tilde{G}$, using the state left by $\mathcal{R}.\text{Sim}$.

We do not separately require $\mathcal{R}.\text{RO}$ to behave like a random oracle. The advantage compares the complete experiments, including the output predicate's queries to $\tilde{G}$ in the simulation game.

7.2 Achievability Results in the ROM and QROM

We now prove that distributional injective extractability holds under the ROM definition above for every Γ -PI wrapping triple, and under the QROM definition above for the FO wrapping triple, even relative to oblivious sampling.

The main condition is that $\Gamma(\lambda, t)/2^\ell$ is negligible for every polynomial t . For the FO wrapping triple, this means that every ciphertext of polynomial length is valid under only a negligible fraction of the symmetric keys. Since SE has no oracle access, this condition follows from INT-CTXT security against non-uniform adversaries (Lemma 2.11).

Proposition 7.3 (1-DINJEXT2 relative to O_{samp} in the ROM). *Let $(\mathcal{G}, \text{Param}, P)$ be a wrapping triple with $\mathcal{G}$ Γ -PI. Let $\mathcal{B}$ be a PPT adversary in the ROM that makes at most q_{inv} queries to its inversion oracle, each on an image of length at most t_{max} . Then there is a PPT simulator $\mathcal{R}$ that meets the condition on challenge queries of Definition 5.5 for $\mathcal{B}$ and satisfies*

$$\text{Adv}_{\text{RO}, \mathcal{G}, \text{Param}, P}^{1\text{-DINJEXT2-samp}}(\mathcal{B}, \mathcal{R}) \leq q_{\text{inv}} \cdot \frac{\Gamma(\lambda, t_{\text{max}})}{2^\ell}.$$

The proof is in Appendix C.1. For the FO wrapping triple, the bound is $q_{\text{inv}} \cdot \Gamma_{\text{SE}}(\lambda, t_{\text{max}})/2^{\ell_K}$.

Theorem 7.4 (1-DINJEXT2 relative to O_{samp} in the QROM). *Let $\mathcal{B}$ be a QPT adversary in the QROM that makes at most q_{inv} queries to its inversion oracle, each on an image of length at most t_{max} . Then there is a QPT simulator $\mathcal{R}$ that meets the condition on challenge queries of Definition 5.5 for $\mathcal{B}$ and satisfies*

$$\text{Adv}_{\text{QRO}, \text{Wrap}_{\text{FO}}}^{1\text{-DINJEXT2-samp}}(\mathcal{B}, \mathcal{R}) \leq q_{\text{inv}} \left(2\sqrt{\frac{\Gamma_{\text{SE}}(\lambda, t_{\text{max}})}{2^{\ell_K}}} + \frac{1}{2^n} \right).$$

The proof is in Appendix C.2.

The term $1/2^n$ bounds the probability that an inversion query made before the challenge query has preimage x^* . Our simulator returns $\perp$ on such a query, because the game answers the queries at x^* . The term is negligible by our standing assumption on n .

References

- [1] Abdalla, M., Bellare, M., Neven, G.: Robust encryption. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 480–497. Springer, Berlin, Heidelberg (Feb 2010). https://doi.org/10.1007/978-3-642-11799-2_28
- [2] Alapati, N., Maram, V., Masny, D.: Non-observable quantum random oracle model. In: Johansson, T., Smith-Tone, D. (eds.) PQCrypto 2023. pp. 417–444. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-40003-2_16
- [3] Albertini, A., Duong, T., Gueron, S., Kölbl, S., Luykx, A., Schmieg, S.: How to abuse and fix authenticated encryption without key commitment. In: Butler, K.R.B., Thomas, K. (eds.) USENIX Security 2022. pp. 3291–3308. USENIX Association (Aug 2022), <https://www.usenix.org/conference/usenixsecurity22/presentation/albertini>

- [4] Albrecht, M.R., Bernstein, D.J., Chou, T., Cid, C., Gilcher, J., Lange, T., Maram, V., von Maurich, I., Misoczki, R., Niederhagen, R., Paterson, K.G., Persichetti, E., Peters, C., Schwabe, P., Sendrier, N., Szefer, J., Tjhai, C.J., Tomlinson, M., Wang, W.: Classic McEliece. Tech. rep., National Institute of Standards and Technology (2022), available at <https://csrc.nist.gov/projects/post-quantum-cryptography/round-4-submissions>
- [5] Ananth, P., Bhaskar, R.: Non observability in the random oracle model. In: Susilo, W., Reyhanitabar, R. (eds.) ProvSec 2013. LNCS, vol. 8209, pp. 86–103. Springer, Berlin, Heidelberg (Oct 2013). https://doi.org/10.1007/978-3-642-41227-1_5
- [6] Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. J. ACM **59**(2), 6:1–6:48 (2012). <https://doi.org/10.1145/2160158.2160159>, <https://doi.org/10.1145/2160158.2160159>
- [7] Bellare, M., Boldyreva, A., O’Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Berlin, Heidelberg (Aug 2007). https://doi.org/10.1007/978-3-540-74143-5_30
- [8] Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: Krawczyk, H. (ed.) CRYPTO’98. LNCS, vol. 1462, pp. 26–45. Springer, Berlin, Heidelberg (Aug 1998). <https://doi.org/10.1007/BFb0055718>
- [9] Bellare, M., Hoang, V.T.: Efficient schemes for committing authenticated encryption. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 845–875. Springer, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-07085-3_29
- [10] Bellare, M., Hoang, V.T., Keelveedhi, S.: Instantiating random oracles via UCEs. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 398–415. Springer, Berlin, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40084-1_23
- [11] Bellare, M., Hofheinz, D., Yilek, S.: Possibility and impossibility results for encryption and commitment secure under selective opening. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 1–35. Springer, Berlin, Heidelberg (Apr 2009). https://doi.org/10.1007/978-3-642-01001-9_1
- [12] Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Berlin, Heidelberg (Dec 2000). https://doi.org/10.1007/3-540-44448-3_41
- [13] Bellare, M., Palacio, A.: Towards plaintext-aware public-key encryption without random oracles. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 48–62. Springer, Berlin, Heidelberg (Dec 2004). https://doi.org/10.1007/978-3-540-30539-2_4
- [14] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 93. pp. 62–73. ACM Press (Nov 1993). <https://doi.org/10.1145/168588.168596>

- [15] Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: De Santis, A. (ed.) EUROCRYPT'94. LNCS, vol. 950, pp. 92–111. Springer, Berlin, Heidelberg (May 1995). <https://doi.org/10.1007/BFb0053428>
- [16] Bernstein, D.J., Lange, T.: Non-uniform cracks in the concrete: The power of free precomputation. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 321–340. Springer, Berlin, Heidelberg (Dec 2013). https://doi.org/10.1007/978-3-642-42045-0_17
- [17] Bernstein, D.J., Persichetti, E.: Towards KEM unification. Cryptology ePrint Archive, Report 2018/526 (2018), <https://eprint.iacr.org/2018/526>
- [18] Bindel, N., Hamburg, M., Hövelmanns, K., Hülsing, A., Persichetti, E.: Tighter proofs of CCA security in the quantum random oracle model. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019, Part II. LNCS, vol. 11892, pp. 61–90. Springer, Cham (Dec 2019). https://doi.org/10.1007/978-3-030-36033-7_3
- [19] Bitansky, N., Canetti, R., Paneth, O., Rosen, A.: On the existence of extractable one-way functions. In: Shmoys, D.B. (ed.) 46th ACM STOC. pp. 505–514. ACM Press (May / Jun 2014). <https://doi.org/10.1145/2591796.2591859>
- [20] Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 280–300. Springer, Berlin, Heidelberg (Dec 2013). https://doi.org/10.1007/978-3-642-42045-0_15
- [21] Bos, J.W., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS – kyber: A cca-secure module-lattice-based KEM. In: 2018 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 353–367. IEEE (2018)
- [22] Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 337–367. Springer, Berlin, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46803-6_12
- [23] Branco, P., Döttling, N., Jain, A., Malavolta, G., Mathialagan, S., Peters, S., Vaikuntanathan, V.: Pseudorandom obfuscation and applications. In: Kalai, Y.T., Kamara, S.F. (eds.) CRYPTO 2025, Part V. LNCS, vol. 16004, pp. 663–698. Springer, Cham (Aug 2025). https://doi.org/10.1007/978-3-032-01901-1_21
- [24] Brzuska, C., Farshim, P., Mittelbach, A.: Indistinguishability obfuscation and UCEs: The case of computationally unpredictable sources. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 188–205. Springer, Berlin, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44371-2_11
- [25] Brzuska, C., Farshim, P., Mittelbach, A.: Random-oracle uninstantiability from indistinguishability obfuscation. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 428–455. Springer, Berlin, Heidelberg (Mar 2015). https://doi.org/10.1007/978-3-662-46497-7_17
- [26] Brzuska, C., Mittelbach, A.: Using indistinguishability obfuscation via UCEs. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part II. LNCS, vol. 8874, pp. 122–141. Springer, Berlin, Heidelberg (Dec 2014). https://doi.org/10.1007/978-3-662-45608-8_7

- [27] Canetti, R.: Towards realizing random oracles: Hash functions that hide all partial information. In: Kaliski, Jr., B.S. (ed.) CRYPTO'97. LNCS, vol. 1294, pp. 455–469. Springer, Berlin, Heidelberg (Aug 1997). <https://doi.org/10.1007/BFb0052255>
- [28] Canetti, R., Chen, Y., Holmgren, J., Lombardi, A., Rothblum, G.N., Rothblum, R.D., Wichs, D.: Fiat-Shamir: from practice to theory. In: Charikar, M., Cohen, E. (eds.) 51st ACM STOC. pp. 1082–1090. ACM Press (Jun 2019). <https://doi.org/10.1145/3313276.3316380>
- [29] Canetti, R., Chen, Y., Reyzin, L.: On the correlation intractability of obfuscated pseudorandom functions. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A, Part I. LNCS, vol. 9562, pp. 389–415. Springer, Berlin, Heidelberg (Jan 2016). https://doi.org/10.1007/978-3-662-49096-9_17
- [30] Canetti, R., Dakdouk, R.R.: Extractable perfectly one-way functions. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 449–460. Springer, Berlin, Heidelberg (Jul 2008). https://doi.org/10.1007/978-3-540-70583-3_37
- [31] Canetti, R., Dakdouk, R.R.: Towards a theory of extractable functions. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 595–613. Springer, Berlin, Heidelberg (Mar 2009). https://doi.org/10.1007/978-3-642-00457-5_35
- [32] Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM **51**(4), 557–594 (2004)
- [33] Canetti, R., Lin, H., Tessaro, S., Vaikuntanathan, V.: Obfuscation of probabilistic circuits and applications. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 468–497. Springer, Berlin, Heidelberg (Mar 2015). https://doi.org/10.1007/978-3-662-46497-7_19
- [34] Cao, N., O'Neill, A., Zaheri, M.: Toward RSA-OAEP without random oracles. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part I. LNCS, vol. 12110, pp. 279–308. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45374-9_10
- [35] Chen, C., Danba, O., Hoffstein, J., Hulsing, A., Rijneveld, J., Schanck, J.M., Schwabe, P., Whyte, W., Zhang, Z., Saito, T., Yamakawa, T., Xagawa, K.: NTRU. Tech. rep., National Institute of Standards and Technology (2020), available at <https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions>
- [36] Chen, Y., Mao, X.: Universal computational extractors and multi-bit AIPO from lattice assumptions. In: Fehr, S., Fouque, P.A. (eds.) EUROCRYPT 2025, Part III. LNCS, vol. 15603, pp. 63–93. Springer, Cham (May 2025). https://doi.org/10.1007/978-3-031-91131-6_3
- [37] Damgård, I.: Towards practical public key systems secure against chosen ciphertext attacks. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 445–456. Springer, Berlin, Heidelberg (Aug 1992). https://doi.org/10.1007/3-540-46766-1_36
- [38] D'Anvers, J.P., Guo, Q., Johansson, T., Nilsson, A., Vercauteren, F., Verbauwheide, I.: Decryption failure attacks on IND-CCA secure lattice-based schemes. In: Lin, D., Sako, K. (eds.) PKC 2019, Part II. LNCS, vol. 11443, pp. 565–598. Springer, Cham (Apr 2019). https://doi.org/10.1007/978-3-030-17259-6_19

- [39] D’Anvers, J.P., Rossi, M., Virdia, F.: (One) failure is not an option: Bootstrapping the search for failures in lattice-based encryption schemes. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part III. LNCS, vol. 12107, pp. 3–33. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45727-3_1
- [40] Dodis, Y., Ristenpart, T., Shrimpton, T.: Salvaging Merkle-Damgård for practical applications. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 371–388. Springer, Berlin, Heidelberg (Apr 2009). https://doi.org/10.1007/978-3-642-01001-9_22
- [41] Don, J., Fehr, S., Majenz, C., Schaffner, C.: Online-extractability in the quantum random-oracle model. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part III. LNCS, vol. 13277, pp. 677–706. Springer, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-07082-2_24
- [42] Ebrahimi, E., van Wier, J.: Post-quantum plaintext-awareness. In: Cheon, J.H., Johansson, T. (eds.) PQCrypto 2022. pp. 260–285. Springer, Cham (Sep 2022). https://doi.org/10.1007/978-3-031-17234-2_13
- [43] Farshim, P., Orlandi, C., Roşie, R.: Security of symmetric primitives under incorrect usage of keys. IACR Trans. Symm. Cryptol. **2017**(1), 449–473 (2017). <https://doi.org/10.13154/tosc.v2017.i1.449-473>
- [44] Fiedler, R., Langrehr, R.: On deniable authentication against malicious verifiers. In: Kalai, Y.T., Kamara, S.F. (eds.) CRYPTO 2025, Part VIII. LNCS, vol. 16007, pp. 3–38. Springer, Cham (Aug 2025). https://doi.org/10.1007/978-3-032-01913-4_1
- [45] Fischlin, M., Rohrbach, F.: Searching for ELFs in the cryptographic forest. In: Rothblum, G.N., Wee, H. (eds.) TCC 2023, Part III. LNCS, vol. 14371, pp. 207–236. Springer, Cham (Nov / Dec 2023). https://doi.org/10.1007/978-3-031-48621-0_8
- [46] Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. Journal of Cryptology **26**(1), 80–101 (Jan 2013). <https://doi.org/10.1007/s00145-011-9114-1>
- [47] Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. SIAM J. Comput. **45**(3), 882–929 (2016). <https://doi.org/10.1137/14095772X>, <https://doi.org/10.1137/14095772X>
- [48] Garg, S., Gentry, C., Halevi, S., Wichs, D.: On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 518–535. Springer, Berlin, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44371-2_29
- [49] Goyal, R., Koppula, V., Waters, B.: Lockable obfuscation. In: Umans, C. (ed.) 58th FOCS. pp. 612–621. IEEE Computer Society Press (Oct 2017). <https://doi.org/10.1109/FOCS.2017.62>
- [50] Grubbs, P., Maram, V., Paterson, K.G.: Anonymous, robust post-quantum public key encryption. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part III. LNCS, vol. 13277, pp. 402–432. Springer, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-07082-2_15

- [51] Hada, S., Tanaka, T.: On the existence of 3-round zero-knowledge protocols. In: Krawczyk, H. (ed.) CRYPTO'98. LNCS, vol. 1462, pp. 408–423. Springer, Berlin, Heidelberg (Aug 1998). <https://doi.org/10.1007/BFb0055744>
- [52] Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. *SIAM Journal on Computing* **28**(4), 1364–1396 (1999)
- [53] Hoang, V.T., Krovetz, T., Rogaway, P.: Robust authenticated-encryption AEZ and the problem that it solves. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 15–44. Springer, Berlin, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46800-5_2
- [54] Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 341–371. Springer, Cham (Nov 2017). https://doi.org/10.1007/978-3-319-70500-2_12
- [55] Hohenberger, S., Sahai, A., Waters, B.: Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 201–220. Springer, Berlin, Heidelberg (May 2014). https://doi.org/10.1007/978-3-642-55220-5_12
- [56] Hövelmanns, K., Hülsing, A., Majenz, C.: Failing gracefully: Decryption failures and the Fujisaki-Okamoto transform. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part IV. LNCS, vol. 13794, pp. 414–443. Springer, Cham (Dec 2022). https://doi.org/10.1007/978-3-031-22972-5_15
- [57] Hövelmanns, K., Kiltz, E., Schäge, S., Unruh, D.: Generic authenticated key exchange in the quantum random oracle model. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part II. LNCS, vol. 12111, pp. 389–422. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45388-6_14
- [58] Jiang, H., Zhang, Z., Chen, L., Wang, H., Ma, Z.: IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisited. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 96–125. Springer, Cham (Aug 2018). https://doi.org/10.1007/978-3-319-96878-0_4
- [59] Jiang, H., Zhang, Z., Ma, Z.: On the non-tightness of measurement-based reductions for key encapsulation mechanism in the quantum random oracle model. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part I. LNCS, vol. 13090, pp. 487–517. Springer, Cham (Dec 2021). https://doi.org/10.1007/978-3-030-92062-3_17
- [60] Kalai, Y.T., Rothblum, G.N., Rothblum, R.D.: From obfuscation to the security of Fiat-Shamir for proofs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II. LNCS, vol. 10402, pp. 224–251. Springer, Cham (Aug 2017). https://doi.org/10.1007/978-3-319-63715-0_8
- [61] Khovratovich, D., Rothblum, R.D., Soukhanov, L.: How to prove false statements: Practical attacks on Fiat-Shamir. In: Kalai, Y.T., Kamara, S.F. (eds.) CRYPTO 2025, Part VI. LNCS, vol. 16005, pp. 3–26. Springer, Cham (Aug 2025). https://doi.org/10.1007/978-3-032-01887-8_1

- [62] Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013. pp. 669–684. ACM Press (Nov 2013). <https://doi.org/10.1145/2508859.2516668>
- [63] Kiltz, E., O’Neill, A., Smith, A.: Instantiability of RSA-OAEP under chosen-plaintext attack. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 295–313. Springer, Berlin, Heidelberg (Aug 2010). https://doi.org/10.1007/978-3-642-14623-7_16
- [64] Matsuda, T., Hanaoka, G.: Chosen ciphertext security via UCE. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 56–76. Springer, Berlin, Heidelberg (Mar 2014). https://doi.org/10.1007/978-3-642-54631-0_4
- [65] Murphy, A., O’Neill, A., Zaheri, M.: Instantiability of classical random-oracle-model encryption transforms. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part IV. LNCS, vol. 13794, pp. 323–352. Springer, Cham (Dec 2022). https://doi.org/10.1007/978-3-031-22972-5_12
- [66] National Institute of Standards and Technology: Module-Lattice-Based Key-Encapsulation Mechanism Standard. Federal Information Processing Standards Publication FIPS PUB 203, U.S. Department of Commerce, Washington, D.C. (Aug 2024). <https://doi.org/10.6028/NIST.FIPS.203>, <https://doi.org/10.6028/NIST.FIPS.203>
- [67] Nielsen, J.B.: Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 111–126. Springer, Berlin, Heidelberg (Aug 2002). https://doi.org/10.1007/3-540-45708-9_8
- [68] Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 187–196. ACM Press (May 2008). <https://doi.org/10.1145/1374376.1374406>
- [69] Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. SIAM Journal on Computing 40(6), 1803–1844 (2011). <https://doi.org/10.1137/080733954>
- [70] Saito, T., Xagawa, K., Yamakawa, T.: Tightly-secure key-encapsulation mechanism in the quantum random oracle model. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 520–551. Springer, Cham (Apr / May 2018). https://doi.org/10.1007/978-3-319-78372-7_17
- [71] Targhi, E.E., Unruh, D.: Post-quantum security of the Fujisaki-Okamoto and OAEP transforms. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 192–216. Springer, Berlin, Heidelberg (Oct / Nov 2016). https://doi.org/10.1007/978-3-662-53644-5_8
- [72] Wichs, D., Zirdelis, G.: Obfuscating compute-and-compare programs under LWE. In: Umans, C. (ed.) 58th FOCS. pp. 600–611. IEEE Computer Society Press (Oct 2017)
- [73] Zhandry, M.: The magic of ELFs. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 479–508. Springer, Berlin, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53018-4_18
- [74] Zhandry, M.: How to record quantum queries, and applications to quantum indistinguishability. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 239–268. Springer, Cham (Aug 2019). https://doi.org/10.1007/978-3-030-26951-7_9

- [75] Zhandry, M.: Augmented random oracles. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part III. LNCS, vol. 13509, pp. 35–65. Springer, Cham (Aug 2022). https://doi.org/10.1007/978-3-031-15982-4_2
- [76] Zhandry, M.: Quantum money from abelian group actions. In: Guruswami, V. (ed.) ITCS 2024. vol. 287, pp. 101:1–101:23. LIPIcs (Jan / Feb 2024). <https://doi.org/10.4230/LIPIcs.ITCS.2024.101>, full version: <https://eprint.iacr.org/2023/1097>

A Modular Composition for DINJEXT and UCE

Theorem 6.2 requires one hash function family that is both UCE-secure for the source S_F and 1-DINJEXT2[Γ_{SE}] for the FO wrapping triple. The first property follows from S_1^{scup} -UCE security when F is one-way. We show that the two properties can be obtained modularly from two separate hash function families; thus, research can focus on each property separately, subject to the conditions of Theorem A.4. Let G_{EXT} be a family that provides extraction and G_{UCE} a family that provides UCE security, and let G be their cascade

$$G((k_1, k_2), x) \stackrel{\text{def}}{=} G_{\text{UCE}}(k_1, G_{\text{EXT}}(k_2, x)).$$

Theorem A.4 says that G has both properties, assuming extraction of G_{EXT} for a derived wrapping triple that contains G_{UCE} , one further assumption on G_{EXT} (Definition A.2), and nothing beyond S_1^{scup} -UCE security of G_{UCE} .

The cascade. Let G_{UCE} and G_{EXT} be keyed hash function families, with key generation algorithms $G_{\text{UCE}}.\text{Kg}$ and $G_{\text{EXT}}.\text{Kg}$, key spaces $\mathcal{K}_1$ and $\mathcal{K}_2$, and evaluation maps

$$G_{\text{UCE}}(k_1, \cdot) : \{0, 1\}^s \rightarrow \{0, 1\}^\ell \quad \text{and} \quad G_{\text{EXT}}(k_2, \cdot) : \{0, 1\}^* \rightarrow \{0, 1\}^s.$$

The cascade G has key space $\mathcal{K}_1 \times \mathcal{K}_2$. Its key generation $G.\text{Kg}(1^\lambda)$ samples $k_1 \leftarrow \$ G_{\text{UCE}}.\text{Kg}(1^\lambda)$ and $k_2 \leftarrow \$ G_{\text{EXT}}.\text{Kg}(1^\lambda)$ independently and returns (k_1, k_2) . Its evaluation is the cascade displayed above.

The derived wrapping triple. Fix a Γ -PI wrapping function $\mathcal{G} = \{g_p\}_{p \in \mathcal{P}}$ with y -input length ℓ , a setup sampler Param for $\mathcal{G}$, and a preimage creator P for $(\mathcal{G}, G)$. The *derived wrapping function* is $\tilde{\mathcal{G}} = \{\tilde{g}_{\tilde{p}}\}_{\tilde{p} \in \tilde{\mathcal{P}}}$ with parameter set $\tilde{\mathcal{P}} = \mathcal{K}_1 \times \mathcal{P}$, preimage domain $\tilde{\mathcal{X}} = \mathcal{X}$, y -input length s , and

$$\tilde{g}_{(k_1, p)}(x, h, w; \rho) \stackrel{\text{def}}{=} g_p(x, G_{\text{UCE}}(k_1, h), w; \rho).$$

Its auxiliary-input space is $\mathcal{W}$ and its coin space is $\mathcal{R}$. The derived preimage creator has the same challenge-input space $\mathcal{W}'$ as P . The derived setup sampler and preimage creator are

$$\begin{aligned} \widetilde{\text{Param}}(1^\lambda) : \quad & k_1 \leftarrow \$ G_{\text{UCE}}.\text{Kg}(1^\lambda); \quad (p, \text{aux}) \leftarrow \$ \text{Param}(1^\lambda); \\ & \text{return } ((k_1, p), \text{aux}); \end{aligned}$$

$$\tilde{P}(k_2, (k_1, p), \text{aux}, w') \stackrel{\text{def}}{=} P((k_1, k_2), p, \text{aux}, w').$$

The setup sampler puts the key k_1 of the outer hash into the public wrapper parameter. The derived preimage creator $\tilde{P}$ reassembles the two keys and calls P .

Definition A.1 (Preimage multiplicity). A keyed function family H with evaluation maps $H(k, \cdot) : \mathcal{X} \rightarrow \mathcal{Y}$ has preimage multiplicity at most μ if

$$|\{x \in \mathcal{X} : H(k, x) = y\}| \leq \mu$$

for every key $k \in [H.\text{Kg}(1^\lambda)]$ and every $y \in \mathcal{Y}$.

Definition A.2 (Unpredictability-preserving). A keyed function family H with evaluation maps $H(k, \cdot) : \{0, 1\}^* \rightarrow \{0, 1\}^s$ is unpredictability-preserving if for every PPT sampler Samp and every PPT predictor Q there are a PPT predictor Q' and a polynomial q such that

$$\Pr[Q(k, z) = H(k, x)] \leq q(\lambda) \cdot \Pr[Q'(z) = x] + \text{negl}(\lambda),$$

where $(x, z) \leftarrow \$\text{Samp}(1^\lambda)$ and $k \leftarrow \$H.\text{Kg}(1^\lambda)$ are sampled independently. The post-quantum version quantifies over QPT predictors Q and Q' .

Discussion. Preimage multiplicity of G_{UCE} bounds how much the y -ambiguity grows in the derived triple (Proposition A.3). Unpredictability preservation (Definition A.2) is the additional assumption that the composition requires. It says that a hash query which is hard to predict stays hard to predict after G_{EXT} is applied to it. The sampler does not receive k , so the pair (x, z) is independent of the key.

Proposition A.3 (Structure of the derived wrapping triple). Let $\mathcal{G}$ be Γ -PI and let G_{UCE} have preimage multiplicity at most μ . Then $\tilde{\mathcal{G}}$ is $\mu\Gamma$ -PI, where $\mu\Gamma$ denotes the bound $(\lambda, t) \mapsto \mu \cdot \Gamma(\lambda, t)$.

Proof. Evaluation is efficient by composition, and the preimage-length bound is inherited from $\mathcal{G}$. Fix $\tilde{p} = (k_1, p)$. The consistency predicate of $\tilde{g}_{\tilde{p}}$ is zero when $h \notin \{0, 1\}^s$; otherwise it is

$$\begin{aligned} \tilde{C}_{\tilde{p}}(x, h, z) = 1 &\iff x \in \mathcal{X} \wedge \exists(w, \rho) : g_p(x, G_{\text{UCE}}(k_1, h), w; \rho) = z \\ &\iff C_p(x, G_{\text{UCE}}(k_1, h), z) = 1, \end{aligned}$$

where C_p is the consistency predicate of g_p . The predicate $\tilde{C}_{\tilde{p}}$ is efficiently computable because C_p and G_{UCE} are. For x -injectivity, suppose that $\tilde{C}_{\tilde{p}}(x, h, z) = 1$ and $\tilde{C}_{\tilde{p}}(x', h', z) = 1$. Put $y = G_{\text{UCE}}(k_1, h)$ and $y' = G_{\text{UCE}}(k_1, h')$. By the display above, $C_p(x, y, z) = 1$ and $C_p(x', y', z) = 1$, so $x = x'$ by x -injectivity of g_p . For y -ambiguity, fix $x \in \mathcal{X}$ and $z \in \{0, 1\}^*$, and let $Y = \{y \in \{0, 1\}^\ell : C_p(x, y, z) = 1\}$, so that $|Y| \leq \Gamma(\lambda, |z|)$. Then

$$\{h \in \{0, 1\}^s : \tilde{C}_{\tilde{p}}(x, h, z) = 1\} = \{h \in \{0, 1\}^s : G_{\text{UCE}}(k_1, h) \in Y\},$$

and each $y \in Y$ has at most μ preimages under $G_{\text{UCE}}(k_1, \cdot)$. Hence this set has size at most $\mu \cdot \Gamma(\lambda, |z|)$. $\square$

The notions DINJEXT_ι below are those of Definition 5.6, and for $\iota = 2$ this is 1-DINJEXT2 of Definition 5.5.

Theorem A.4 (Modular composition). Let $\mathcal{G}$ be Γ -PI, let Param be a setup sampler for $\mathcal{G}$, let P be a preimage creator for $(\mathcal{G}, G)$, and let $(\tilde{\mathcal{G}}, \widetilde{\text{Param}}, \tilde{P})$ be the derived wrapping triple. Let G_{UCE} have preimage multiplicity at most μ and fix $\iota \in \{0, 1, 2\}$. Assume

1. G_{EXT} satisfies $\text{DINJEXT}_\iota[\mu\Gamma]$ for $(\widetilde{\mathcal{G}}, \widetilde{\text{Param}}, \widetilde{P})$,
2. G_{EXT} is unpredictability-preserving, and
3. G_{UCE} is $\mathcal{S}_1^{\text{scup}}$ -UCE-secure.

Then the cascade G satisfies $\text{DINJEXT}_\iota[\Gamma]$ for $(\mathcal{G}, \text{Param}, P)$ and is $\mathcal{S}_1^{\text{scup}}$ -UCE-secure.

Concretely, for every DINJEXT -adversary $\mathcal{B}$ for G there is a DINJEXT -adversary $\widetilde{\mathcal{B}}$ for G_{EXT} that makes the same oracle queries, and for every distributional simulator $\widetilde{\mathcal{R}}$ for $\widetilde{\mathcal{B}}$ there is a distributional simulator $\mathcal{R}$ for $\mathcal{B}$ with

$$\text{Adv}_{G, \mathcal{G}, \text{Param}, P}^{\text{DINJEXT}_\iota}(\mathcal{B}, \mathcal{R}) = \text{Adv}_{G_{\text{EXT}}, \widetilde{\mathcal{G}}, \widetilde{\text{Param}}, \widetilde{P}}^{\text{DINJEXT}_\iota}(\widetilde{\mathcal{B}}, \widetilde{\mathcal{R}}),$$

and for every single-query UCE adversary (S, D) against G with $S \in \mathcal{S}_1^{\text{scup}}$ there is a single-query UCE adversary (S', D') against G_{UCE} with $S' \in \mathcal{S}_1^{\text{scup}}$ and $\text{Adv}_G^{\text{uce}}((S, D)) = \text{Adv}_{G_{\text{UCE}}}^{\text{uce}}((S', D'))$. For $\iota \in \{0, 1\}$ the preimage creators are ignored. The same holds post-quantum when hypotheses 1 and 3 hold post-quantum and G_{EXT} is unpredictability-preserving against QPT predictors.

Proof. Extraction. Fix $\mathcal{B} = (\mathcal{B}_{\text{run}}, \mathcal{B}_{\text{out}})$. Define $\widetilde{\mathcal{B}}$ as follows. Its run stage, on input $(1^\lambda, k_2, (k_1, p))$, runs $\mathcal{B}_{\text{run}}$ on $(1^\lambda, (k_1, k_2), p)$, forwards its oracle queries unchanged, and returns its transcript. Its output stage, on input $(k_2, (k_1, p), \text{aux}, v)$, returns $\mathcal{B}_{\text{out}}((k_1, k_2), p, \text{aux}, v)$. Given a simulator $\widetilde{\mathcal{R}}$ for $\widetilde{\mathcal{B}}$, define $\mathcal{R}$ to run $\widetilde{\mathcal{R}}$ with the same regrouping of its input and with the same challenge oracle, and to return its transcript.

Call the games of $\mathcal{B}$ against G with the triple $(\mathcal{G}, \text{Param}, P)$ the games for G , and the games of $\widetilde{\mathcal{B}}$ against G_{EXT} with the triple $(\widetilde{\mathcal{G}}, \widetilde{\text{Param}}, \widetilde{P})$ the games for G_{EXT} . We show that the two ideal inversion games are the same experiment and that the two simulation games are the same experiment. The equality of advantages follows.

Setup. The game for G samples its hash key and setup as

$$(k_1, k_2) \leftarrow \$ G.\text{Kg}(1^\lambda), \quad (p, \text{aux}) \leftarrow \$ \text{Param}(1^\lambda),$$

and the game for G_{EXT} samples its hash key and setup as

$$k_2 \leftarrow \$ G_{\text{EXT}}.\text{Kg}(1^\lambda), \quad ((k_1, p), \text{aux}) \leftarrow \$ \widetilde{\text{Param}}(1^\lambda).$$

In both games the values k_1 , k_2 , and (p, aux) are independent, with k_1 from $G_{\text{UCE}}.\text{Kg}$, k_2 from $G_{\text{EXT}}.\text{Kg}$, and (p, aux) from Param . So the setup distributions agree, and $\widetilde{\mathcal{B}}_{\text{run}}$ gives $\mathcal{B}_{\text{run}}$ the same input that the game for G gives it.

Hash values. For every $x \in \mathcal{X}$,

$$\begin{aligned} \widetilde{g}_{(k_1, p)}(x, G_{\text{EXT}}(k_2, x), w; \rho) &= g_p(x, G_{\text{UCE}}(k_1, G_{\text{EXT}}(k_2, x)), w; \rho) \\ &= g_p(x, G((k_1, k_2), x), w; \rho). \end{aligned} \tag{23}$$

Challenge oracle. For $\iota = 2$, on input w' the oracle of the game for G_{EXT} runs $\widetilde{P}$, which by definition runs $P((k_1, k_2), p, \text{aux}, w')$, to obtain (x, w, ρ) . It returns the image $\widetilde{g}_{(k_1, p)}(x, G_{\text{EXT}}(k_2, x), w; \rho)$. By Equation (23) this equals $g_p(x, G((k_1, k_2), x), w; \rho)$, which is what the oracle of the game for G returns on the same (x, w, ρ) . Call this image z^* . Both oracles record (x, z^*) and append z^* to L_{chal} . For $\iota \neq 2$ neither game has a challenge oracle.

Inversion oracle. On input z , both oracles apply the same query-count rule and the same L_{chal} rule. The oracle of the game for G_{EXT} then returns $\perp$ unless there is (x, w, ρ) with $x \in \tilde{\mathcal{X}} = \mathcal{X}$ and $\tilde{g}_{(k_1, \rho)}(x, G_{\text{EXT}}(k_2, x), w; \rho) = z$. By Equation (23) this is the condition $g_p(x, G((k_1, k_2), x), w; \rho) = z$ tested by the oracle of the game for G . So the unique x -preimage of z is the same string in both games, and the same-preimage rule compares it with the same recorded set.

Conclusion. In the two ideal inversion games, $\mathcal{B}_{\text{run}}$ receives the same input and the same oracle answers, so it produces the same transcript distribution. The output predicate of the game for G_{EXT} is by definition the output predicate of the game for G applied to the regrouped inputs. In the two simulation games, $\tilde{\mathcal{R}}$ receives the same input and the same challenge oracle, so it too produces the same transcript distribution, and the output predicates coincide as before. Hence the four acceptance probabilities agree pairwise. For $\iota = 2$, the condition on challenge queries in Definition 5.5 also transfers from $\tilde{\mathcal{R}}$ to $\mathcal{R}$. The ideal inversion games for $\mathcal{B}$ and $\tilde{\mathcal{B}}$ are the same experiment, so $\mathcal{B}$ makes no challenge query with probability 1 exactly when $\tilde{\mathcal{B}}$ does, and $\mathcal{R}$ makes a challenge query exactly when $\tilde{\mathcal{R}}$ does. The same argument applies to QPT algorithms. That $\tilde{\mathcal{G}}$ is $\mu\Gamma$ -PI, as hypothesis 1 requires, is Proposition A.3.

UCE. Let (S, D) be a single-query UCE adversary against G with $S \in \mathcal{S}_1^{\text{scup}}$, and write x^* for the single query of S . Define (S', D') as in Figure 24. The source S' samples a key k_2 for G_{EXT} , runs S , answers the query x^* of S by querying its own oracle at $G_{\text{EXT}}(k_2, x^*)$, and leaks k_2 together with the leakage of S . It makes exactly one query.

Advantage. In the UCE game for G_{UCE} with $\beta = 1$, the answer to the query of S is

$$G_{\text{UCE}}(k_1, G_{\text{EXT}}(k_2, x^*)) = G((k_1, k_2), x^*),$$

and the pair (k_1, k_2) has the distribution of $G.\text{Kg}(1^\lambda)$. So S and D see the UCE game for G with $\beta = 1$. With $\beta = 0$, the answer is a uniform string in $\{0, 1\}^\ell$ independent of everything else, since the random function is evaluated at a single point. So S and D see the UCE game for G with $\beta = 0$. Hence $\text{Adv}_G^{\text{uce}}((S, D)) = \text{Adv}_{G_{\text{UCE}}}^{\text{uce}}((S', D'))$.

Membership of S' in $\mathcal{S}_1^{\text{scup}}$. Fix a PPT predictor Q . In the game $\text{Pred}_{S', Q}^{1\text{-scup}}$ of Figure 5, the single query of S' is $q^* = G_{\text{EXT}}(k_2, x^*)$, the oracle answers it with a uniform $h \leftarrow_{\$} \{0, 1\}^\ell$, and Q receives the view $((k_2, L), h)$, where L is the leakage of S . The game returns 1 if and only if the output of Q equals q^* . Below we write this input of Q as $(k_2, (L, h))$, which is the same information regrouped.

Let $\text{Samp}(1^\lambda)$ sample $h \leftarrow_{\$} \{0, 1\}^\ell$, run S with the constant oracle that answers h , and return $(x^*, (L, h))$, where x^* is the query of S and L its leakage. The sampler does not use k_2 . In the prediction game for S' , the answer h to the single query is uniform and independent of (k_2, x^*) , so the joint distribution of (k_2, x^*, L, h) there is the distribution of a key k_2 from $G_{\text{EXT}}.\text{Kg}(1^\lambda)$ together with an independent output $(x^*, (L, h))$ of $\text{Samp}(1^\lambda)$. Applying Definition A.2 to G_{EXT} with this sampler and the predictor Q gives a PPT predictor Q' and a polynomial q with

$$\Pr[Q(k_2, (L, h)) = G_{\text{EXT}}(k_2, x^*)] \leq q(\lambda) \cdot \Pr[Q'(L, h) = x^*] + \text{negl}(\lambda).$$

The left-hand side is $\Pr[\text{Pred}_{S', Q}^{1\text{-scup}} \Rightarrow 1]$. On the right-hand side, the triple (x^*, L, h) produced by Samp has the distribution of the query, the leakage, and the oracle answer in the game $\text{Pred}_{S, Q'}^{1\text{-scup}}$, whose view is (L, h) . So the right-hand side is $q(\lambda) \cdot \Pr[\text{Pred}_{S, Q'}^{1\text{-scup}} \Rightarrow 1] + \text{negl}(\lambda)$, which is negligible because $S \in \mathcal{S}_1^{\text{scup}}$. Hence $S' \in \mathcal{S}_1^{\text{scup}}$. For QPT D and QPT predictors the same argument applies, using the post-quantum form of Definition A.2.

□

$\underline{S' \mathcal{O}(1^\lambda)}$ $k_2 \leftarrow \$ G_{\text{EXT}}.\text{Kg}(1^\lambda)$ $L \leftarrow \$ S^{\text{HASH}}(1^\lambda)$ Return (k_2, L) $\underline{\text{HASH}(x)}$ Return $\mathcal{O}(G_{\text{EXT}}(k_2, x))$	$\underline{D'((k_2, L), k_1)}$ Return $D(L, (k_1, k_2))$
---	--

Figure 24: The UCE adversary (S', D') against G_{UCE} built from (S, D) .

The FO instance. For the FO triple Wrap_{FO} of Section 6.1, the derived wrapping function is

$$\tilde{g}_{(k_1, pk)}(x, h, m; \rho) = \left(\text{F.Eval}(pk, x), \text{SE.Enc}(G_{\text{UCE}}(k_1, h), m; \rho) \right).$$

The symmetric key is derived from the inner hash value by G_{UCE} , and the ambiguity bound is $\mu\Gamma_{\text{SE}}$. By Theorem A.4, Theorem 6.2 applies to the cascade G whenever G_{EXT} satisfies $1\text{-DINJEXT2}[\mu\Gamma_{\text{SE}}]$ for this derived wrapping triple, G_{EXT} is unpredictability-preserving, G_{UCE} is $\mathcal{S}_1^{\text{scup}}$ -UCE-secure, F is one-way, and the remaining assumptions on F and SE in that theorem hold. The convention of Section 6 that the keys of G have a fixed encoding length holds for the cascade when it holds for G_{UCE} and G_{EXT} .

Discussion. The theorem moves an assumption; it does not remove one. Extraction is assumed of G_{EXT} for the derived triple, which contains the outer hash G_{UCE} , and the theorem moves that assumption, together with unpredictability preservation, to the cascade. It does not say that extraction for the derived triple is any easier to achieve than for the original one.

The ambiguity parameter changes under the composition. If G_{UCE} maps s -bit strings to ℓ -bit strings with preimage multiplicity at most μ , the derived wrapper is $\mu\Gamma$ -PI with s -bit hash values, so the ratio that governs extraction for G_{EXT} is

$$\frac{\mu\Gamma}{2^s} = \left(\frac{\mu 2^\ell}{2^s} \right) \cdot \frac{\Gamma}{2^\ell}.$$

A negligible ratio $\Gamma/2^\ell$ for the original wrapper is therefore preserved whenever the factor $\mu 2^\ell / 2^s$ is polynomially bounded, for instance when G_{UCE} is regular, and this is a property of G_{UCE} that UCE security does not by itself provide.

B Negative Results for 1-DINJEXT2

In this appendix we give the four counterexamples mentioned in Section 5.5. In each, we build from a given hash function family, under the assumptions stated in each result, a wrapping triple for which that family does not satisfy $1\text{-DINJEXT2}[1]$. The four use different channels to pass “dangerous data”, namely the wrapper input, the challenge witness, the public setup, and the challenge image. The input-channel and setup-channel counterexamples are proved for the weakest notion $\text{DINJEXT0}[1]$, and by Proposition 5.7 they also work for $1\text{-DINJEXT2}[1]$ for every preimage creator. By Proposition 5.8, each counterexample to a distributional notion works for the corresponding coin-based notion.

Conventions. Throughout this appendix, $\text{Hash} = (\text{Hash.Gen}, \text{Hash.Eval})$ is a keyed hash function family with ℓ -bit outputs, and we write H_{hk} for $\text{Hash.Eval}(\text{hk}, \cdot)$. The hash keys output by $\text{Hash.Gen}(1^\lambda)$ lie in a finite set $\mathcal{K}_\lambda \subseteq \{0, 1\}^{\kappa(\lambda)}$ for a polynomial κ , and membership in $\mathcal{K}_\lambda$ is decidable in polynomial time. All computational assumptions are against non-uniform PPT adversaries.

Obfuscation and punctured PRFs. Two of the counterexamples use indistinguishability obfuscation [47] and punctured PRFs [20, 22, 62] with their standard definitions. We write $iO(C; s)$ for the obfuscation of a circuit C with coins s , $K\{\xi\}$ for a PRF key K punctured at the point ξ , and ϵ_{iO} and $\text{Adv}_{\text{PRF}}^{\text{PPRF}}$ for the respective advantages.

B.1 Input Channel Counterexample

If the wrapper input may contain a hash key, the wrapping function can evaluate the hash under that key and cancel the hash value it is given. We show that this already breaks $\text{DINJEXT0}[1]$, with no auxiliary input, no wrapping coins, and no obfuscation.

Counterexample wrapping triple. Let $F = (F.\text{Kg}, F.\text{Eval})$ be a keyed injective one-way function whose domain $F.\text{Dom}_\lambda$ can be sampled uniformly, has polynomial-time membership, and contains only polynomial-length strings. The setup sampler

$$\text{Param}_\perp(1^\lambda) : \quad pk \leftarrow \$ F.\text{Kg}(1^\lambda); \quad a \leftarrow \$ F.\text{Dom}_\lambda; \quad \text{return } ((pk, F.\text{Eval}(pk, a)), \epsilon)$$

publishes the image $u_0 = F.\text{Eval}(pk, a)$ of a uniform point a and discards a . For $p = (pk, u_0)$, the wrapping function $\mathcal{G}_\perp = \{g_{\perp, p}\}$ has preimage domain $\mathcal{X} = F.\text{Dom}_\lambda \times \mathcal{K}_\lambda$ and is defined by

$$g_{\perp, p}((a, K'), y) \stackrel{\text{def}}{=} (F.\text{Eval}(pk, a), K', y \oplus H_{K'}(\langle a, K' \rangle)),$$

where $\langle a, K' \rangle$ is the canonical encoding of the preimage (a, K') , on which the games also evaluate the hash. Its consistency predicate parses $x = (a, K')$ and $z = (u, K, t)$, rejects if parsing fails, $x \notin \mathcal{X}$, or $y \notin \{0, 1\}^\ell$, and otherwise accepts exactly when $F.\text{Eval}(pk, a) = u$, $K' = K$, and $t = y \oplus H_{K'}(\langle a, K' \rangle)$. An image determines x , because $F.\text{Eval}(pk, \cdot)$ is injective and K' appears in the image, and together with x it determines y . So $\mathcal{G}_\perp$ is 1-PI.

Proposition B.1 (Input-channel counterexample). *There is a PPT DINJEXT0 adversary $\mathcal{B}$ such that for every PPT simulator $\mathcal{R}$*

$$\text{Adv}_{\text{Hash}, \mathcal{G}_\perp, \text{Param}_\perp}^{\text{DINJEXT0}}(\mathcal{B}, \mathcal{R}) \geq 1 - \text{negl}(\lambda), \quad (24)$$

and there is a PPT INJEXT0 adversary $\mathcal{A}$ such that for every PPT extractor $\mathcal{E}$

$$\text{Adv}_{\text{Hash}, \mathcal{G}_\perp, \text{Param}_\perp}^{\text{INJEXT0}}(\mathcal{A}, \mathcal{E}) \geq 1 - \text{negl}(\lambda). \quad (25)$$

Proof. The run stage of $\mathcal{B}$ queries its inversion oracle on $z = (u_0, \text{hk}, 0^\ell)$ and returns (z, x) , where x is the reply. The output predicate accepts exactly when $z = (u_0, \text{hk}, 0^\ell)$, x parses as (a', hk) with $a' \in F.\text{Dom}_\lambda$, and $F.\text{Eval}(pk, a') = u_0$. In the ideal inversion game the two hash values cancel,

$$g_{\perp, p}((a, \text{hk}), H_{\text{hk}}(\langle a, \text{hk} \rangle)) = (u_0, \text{hk}, 0^\ell),$$

so the inverter returns (a, hk) and the predicate accepts with probability 1.

In the simulation game, an accepting output contains a preimage of u_0 . So a simulator $\mathcal{R}$ gives an inverter for F , which on input (pk, u_0) samples $hk \leftarrow \$ \text{Hash.Gen}(1^\lambda)$, runs $\mathcal{R}(1^\lambda, hk, (pk, u_0))$, and outputs a' if the reported x parses as (a', hk) . The inputs of $\mathcal{R}$ have the same distribution as in the simulation game, so by one-wayness of F the simulation game accepts with negligible probability. This proves Equation (24), and Equation (25) follows from Proposition 5.8. $\square$

A class of wrapping triples whose input length is bounded below the hash-key length excludes this counterexample, and it contains the FO wrapping triple when the preimage length n of the trapdoor function is below that bound. The remaining counterexamples show that such a bound is not enough on its own.

B.2 Witness Channel Counterexample

Here the wrapping function copies its auxiliary input into the image, and a preimage creator that reads the hash key places a second valid image there. The adversary submits this second image for inversion. The output predicate knows the preimage of the second image and checks the reply, so a simulator must invert that image on its own. The wrapper input has bounded length, so the length bound of Appendix B.1 does not exclude this counterexample.

Counterexample wrapping triple. Let $F = (F.\text{Kg}, F.\text{Eval})$ be as in Appendix B.1. The triple $\text{Wrap}_{\text{wit}} = (\mathcal{G}_{\text{wit}}, \text{Param}_{\text{wit}}, P_{\text{wit}})$ consists of

$$\text{Param}_{\text{wit}}(1^\lambda) : \quad pk \leftarrow \$ F.\text{Kg}(1^\lambda); \quad x_0 \leftarrow \$ F.\text{Dom}_\lambda; \quad \text{return } (pk, x_0),$$

the wrapping function

$$g_{pk}^{\text{wit}}(x, y, w) \stackrel{\text{def}}{=} (F.\text{Eval}(pk, x), y, w)$$

with preimage domain $X = F.\text{Dom}_\lambda$ and auxiliary-input space $\mathcal{W} = \{0, 1\}^*$, and the preimage creator

$$P_{\text{wit}}(hk, pk, x_0, w') : \quad x^* \leftarrow \$ F.\text{Dom}_\lambda; \quad z_0 \leftarrow g_{pk}^{\text{wit}}(x_0, H_{hk}(x_0), \varepsilon); \quad \text{return } (x^*, z_0, \varepsilon).$$

So the public parameter is pk , the hidden state is x_0 , and the challenge image is

$$z^* = (F.\text{Eval}(pk, x^*), H_{hk}(x^*), z_0).$$

The consistency predicate accepts (x, y, z) exactly when $x \in F.\text{Dom}_\lambda$, $y \in \{0, 1\}^\ell$, and z parses as (u, t, w) with $F.\text{Eval}(pk, x) = u$ and $y = t$. So $\mathcal{G}_{\text{wit}}$ is 1-PI.

Proposition B.2 (Witness-channel counterexample). *Suppose that every PPT algorithm, given (hk, pk, z_0) for $hk \leftarrow \$ \text{Hash.Gen}(1^\lambda)$, $(pk, x_0) \leftarrow \$ \text{Param}_{\text{wit}}(1^\lambda)$, and $z_0 = g_{pk}^{\text{wit}}(x_0, H_{hk}(x_0), \varepsilon)$, outputs x_0 with negligible probability. Then there is a PPT 1-DINJEXT2 adversary $\mathcal{B}$ such that for every PPT simulator $\mathcal{R}$*

$$\text{Adv}_{\text{Hash}, \text{Wrap}_{\text{wit}}}^{1\text{-DINJEXT2}}(\mathcal{B}, \mathcal{R}) \geq 1 - \text{negl}(\lambda), \quad (26)$$

and there is a PPT 1-INJEXT2 adversary $\mathcal{A}$ such that for every PPT extractor $\mathcal{E}$

$$\text{Adv}_{\text{Hash}, \text{Wrap}_{\text{wit}}}^{1\text{-INJEXT2}}(\mathcal{A}, \mathcal{E}) \geq 1 - \text{negl}(\lambda). \quad (27)$$

Proof. The run stage of $\mathcal{B}$ queries its challenge oracle on ε , reads z_0 from the last component of the challenge image z^* , queries its inversion oracle on z_0 , and returns (z_0, x) , where x is the reply. Given the hidden state x_0 , the output predicate accepts exactly when $x \in \text{F.Dom}_\lambda$ and

$$z_0 = g_{pk}^{\text{wit}}(x_0, H_{hk}(x_0), \varepsilon) = g_{pk}^{\text{wit}}(x, H_{hk}(x), \varepsilon).$$

In the ideal inversion game, $z_0 \notin L_{\text{chal}}$, because the last component of z^* is z_0 while the last component of z_0 is ε . The x -preimage of z_0 is x_0 . So the inverter returns x_0 unless $x_0 = x^*$, in which case it rejects the query as same- x -preimage mauling. The ideal inversion game therefore accepts with probability $1 - 1/|\text{F.Dom}_\lambda|$, which is overwhelming because guessing a uniform point of F.Dom_λ breaks the hypothesis with probability $1/|\text{F.Dom}_\lambda|$.

In the simulation game, an accepting output contains x_0 , because $\text{F.Eval}(pk, \cdot)$ is injective. So a simulator $\mathcal{R}$ gives an algorithm that, on input (hk, pk, z_0) as in the hypothesis, outputs x_0 . It runs $\mathcal{R}(1^\lambda, hk, pk)$, answers the first challenge query by sampling $x^* \leftarrow \text{F.Dom}_\lambda$ and returning $g_{pk}^{\text{wit}}(x^*, H_{hk}(x^*), z_0)$, answers later challenge queries with $\perp$, and outputs the x that $\mathcal{R}$ reports. This is exactly the challenge oracle of the simulation game, because the preimage creator uses x_0 only through z_0 . By the hypothesis, the simulation game therefore accepts with negligible probability. This proves Equation (26), and Equation (27) follows from Proposition 5.8. $\square$

The attack relies on the preimage creator reading the hash key. The FO preimage creator does not read it.

B.3 Setup Channel Counterexample

Here the public setup parameter contains an obfuscated program that, on input a hash key, outputs a valid image whose preimage is hidden. We show that this breaks $\text{DINJEXT0}[1]$ even for a wrapping function of the same two-component form as the FO wrapping function, with input length n . There is no preimage creator, and the program computes images directly rather than by calling the wrapping function.

Counterexample wrapping triple. Let $\text{LT} = (\text{IKg}, \text{LKg}, \text{Eval}, \text{Inv})$ be an almost-always lossy trapdoor function with domain $\{0, 1\}^n$ (Definition 2.9), lossiness $\ell_{\text{loss}} \geq \ell + \omega(\log \lambda)$, and negligible β_{loss} , and write $\Phi_{ek} = \text{Eval}(ek, \cdot)$. Let PRF be a punctured PRF with inputs in $\{0, 1\}^n \times \mathcal{K}_\lambda$ and outputs in $\{0, 1\}^n$. The wrapping function $\mathcal{G}^{\text{LT}} = \{g_p\}_{p \in \mathcal{P}}$ has preimage domain $\mathcal{X} = \{0, 1\}^n$ and is indexed by pairs $p = (ek, \Lambda)$, where ek is an injective-mode evaluation key and Λ is any string. It ignores Λ and computes

$$g_p(x, y) \stackrel{\text{def}}{=} (\Phi_{ek}(x), y).$$

Its consistency predicate accepts (x, y, z) exactly when $x \in \{0, 1\}^n$, $y \in \{0, 1\}^\ell$, and $z = (\Phi_{ek}(x), y)$. Since Φ_{ek} is injective, $\mathcal{G}^{\text{LT}}$ is 1-PI. The setup sampler is

$$\text{Param}^{\text{LT}}(1^\lambda) : (ek, td) \leftarrow \text{IKg}(1^\lambda); \quad K \leftarrow \text{PRF.Kg}(1^\lambda); \quad \text{return } ((ek, iO(D_{ek, K})), \varepsilon),$$

where the circuit

$$D_{ek, K}(q, hk') \stackrel{\text{def}}{=} (\Phi_{ek}(\text{PRF}_K(q, hk')), H_{hk'}(\text{PRF}_K(q, hk')))$$

outputs the image of the preimage $\text{PRF}_K(q, hk')$ under the hash key hk' . It is padded to the size of the circuit D' in the proof below.

Proposition B.3 (Setup-channel counterexample). *Assume that iO is an indistinguishability obfuscator, PRF is a secure punctured PRF, and LT is as above. Then there is a PPT DINJEXT0 adversary $\mathcal{B}$ such that for every PPT simulator $\mathcal{R}$*

$$\text{Adv}_{\text{Hash}, \mathcal{G}^{\text{LT}}, \text{Param}^{\text{LT}}}^{\text{DINJEXT0}}(\mathcal{B}, \mathcal{R}) \geq 1 - \text{negl}(\lambda), \quad (28)$$

and there is a PPT INJEXT0 adversary $\mathcal{A}$ such that for every PPT extractor $\mathcal{E}$

$$\text{Adv}_{\text{Hash}, \mathcal{G}^{\text{LT}}, \text{Param}^{\text{LT}}}^{\text{INJEXT0}}(\mathcal{A}, \mathcal{E}) \geq 1 - \text{negl}(\lambda). \quad (29)$$

Proof. Parse $p = (ek, \Lambda)$. The run stage of $\mathcal{B}$ computes $Z = \Lambda(0^n, \text{hk})$, queries its inversion oracle on Z , and returns $(Z, \hat{x})$, where $\hat{x}$ is the reply. The output predicate accepts exactly when $\hat{x} \in \{0, 1\}^n$, $Z = \Lambda(0^n, \text{hk})$, and $g_p(\hat{x}, H_{\text{hk}}(\hat{x})) = Z$. Let $x^* = \text{PRF}_K(0^n, \text{hk})$. By correctness of iO , $Z = (\Phi_{ek}(x^*), H_{\text{hk}}(x^*)) = g_p(x^*, H_{\text{hk}}(x^*))$, so the ideal inverter returns x^* and the predicate accepts with probability 1.

Now fix a PPT simulator $\mathcal{R}$ with output $(Z', \hat{x})$. If the predicate accepts, then $Z' = \Lambda(0^n, \text{hk})$ and $\Phi_{ek}(\hat{x}) = \Phi_{ek}(x^*)$, so $\hat{x} = x^*$ because ek is an injective-mode key. It therefore suffices to bound the probability p_0 that $\hat{x} = x^*$ in the simulation game G_0 . We change G_0 in three steps and let p_i be the probability that $\hat{x} = x^*$ in G_i . The setup sampler does not receive hk , so every reduction below can sample hk first. Every reduction also knows x^* , so it can check whether $\hat{x} = x^*$.

- In G_1 , let $\xi = (0^n, \text{hk})$ and $K\{\xi\} = \text{PRF.Punc}(K, \xi)$, and let the setup parameter contain $iO(D')$ in place of $iO(D_{ek, K})$. The circuit D' outputs $Z = (\Phi_{ek}(x^*), H_{\text{hk}}(x^*))$ on input ξ , and on every other input (q, hk') it outputs $(\Phi_{ek}(v), H_{\text{hk}'}(v))$ for $v = \text{PRF}_{K\{\xi\}}(q, \text{hk}')$. The circuits D' and $D_{ek, K}$ compute the same function, by punctured-PRF correctness, and have the same size. So $|p_0 - p_1| \leq \epsilon_{iO}$.
- In G_2 , the value x^* is a uniform string in $\{0, 1\}^n$, independent of $K\{\xi\}$. The reduction chooses ξ , receives $K\{\xi\}$ and a value for x^* , and builds D' . So $|p_1 - p_2| \leq \text{Adv}_{\text{PRF}}^{\text{PPRF}}$.
- In G_3 , the key ek is sampled by $\text{LKg}(1^\lambda)$. The reduction receives ek and samples everything else, so $|p_2 - p_3|$ is at most the advantage of a distinguisher between the two modes.

In G_3 , everything that $\mathcal{R}$ receives depends on x^* only through $Z = (\Phi_{ek}(x^*), H_{\text{hk}}(x^*))$, and x^* is uniform on $\{0, 1\}^n$. Except with probability β_{loss} , the map $x \mapsto (\Phi_{ek}(x), H_{\text{hk}}(x))$ takes at most $2^{n-\ell_{\text{loss}}+\ell}$ values, and then any guess for x^* given Z is correct with probability at most $2^{n-\ell_{\text{loss}}+\ell}/2^n = 2^{\ell-\ell_{\text{loss}}}$. So $p_3 \leq \beta_{\text{loss}} + 2^{\ell-\ell_{\text{loss}}}$, which is negligible, and hence so is p_0 . This proves Equation (28), and Equation (29) follows from Proposition 5.8. $\square$

B.4 Challenge Image Counterexample

The last counterexample places the obfuscated program in the challenge image itself. Suppose that from a genuine challenge image anyone can compute a second valid image with a different preimage, and that this preimage is hard to find without the inversion oracle. The ideal inverter inverts the second image, whereas a simulator has to find its preimage on its own. The following proposition states this obstruction in general.

Proposition B.4 (Challenge-image barrier). *Let $\text{Wrap} = (\mathcal{G}, \text{Param}, P)$ be a wrapping triple with $\mathcal{G}$ 1-PI, and fix a challenge input $w_0 \in \mathcal{W}'$. Let Next be a deterministic polynomial-time algorithm and Check a PPT*

algorithm. In the two conditions below, probabilities are over $\text{hk} \leftarrow \$ \text{Hash.Gen}(1^\lambda)$, $(p, \text{aux}) \leftarrow \$ \text{Param}(1^\lambda)$, and all coins, and $\mathcal{O}_{\text{chal}}$ is the challenge oracle of Figure 12. The predicates C_p and Check reject when an input is $\perp$ or malformed.

1. Let $z_{\text{ch}} \leftarrow \mathcal{O}_{\text{chal}}(w_0)$, let the oracle use the preimage x_{ch} , and let $z = \text{Next}(1^\lambda, \text{hk}, p, z_{\text{ch}})$. Except with negligible probability, $\text{Check}(\text{hk}, p, \text{aux}, z_{\text{ch}}) = 1$ and $C_p(x, H_{\text{hk}}(x), z) = 1$ for some $x \neq x_{\text{ch}}$.
2. Every PPT algorithm $\mathcal{A}^{\mathcal{O}_{\text{chal}}}(1^\lambda, \text{hk}, p)$ has negligible probability of outputting a pair $(\widehat{z}_{\text{ch}}, \widehat{x})$ with $\text{Check}(\text{hk}, p, \text{aux}, \widehat{z}_{\text{ch}}) = 1$ and $C_p(\widehat{x}, H_{\text{hk}}(\widehat{x}), \text{Next}(1^\lambda, \text{hk}, p, \widehat{z}_{\text{ch}})) = 1$.

Then Hash does not satisfy 1-DINJEXT2[1] for Wrap.

Proof. The run stage of the adversary $\mathcal{B}$ queries its challenge oracle on w_0 , receives z_{ch} , computes $z = \text{Next}(1^\lambda, \text{hk}, p, z_{\text{ch}})$, queries its inversion oracle on z , and returns (z_{ch}, x) , where x is the reply. The output predicate accepts exactly when

$$\text{Check}(\text{hk}, p, \text{aux}, z_{\text{ch}}) = 1 \quad \text{and} \quad C_p(x, H_{\text{hk}}(x), \text{Next}(1^\lambda, \text{hk}, p, z_{\text{ch}})) = 1.$$

In the ideal inversion game, suppose that the event of condition 1 occurs, and let $x \neq x_{\text{ch}}$ be the x -preimage of z . Then $z \neq z_{\text{ch}}$, since z_{ch} has x -preimage x_{ch} and x -preimages are unique. So $z \notin L_{\text{chal}}$, and the mauling test does not apply because $x \neq x_{\text{ch}}$. The inverter therefore returns x , and the predicate accepts. So the ideal inversion game accepts with probability $1 - \text{negl}(\lambda)$.

In the simulation game, a simulator $\mathcal{R}$ computes its output $(\widehat{z}_{\text{ch}}, \widehat{x})$ from $(1^\lambda, \text{hk}, p)$ with access to the challenge oracle, and the predicate accepts exactly when this pair satisfies the event of condition 2. So the simulation game accepts with negligible probability, and the advantage of $\mathcal{B}$ is $1 - \text{negl}(\lambda)$ for every PPT simulator. $\square$

A concrete construction. Here the obfuscated program has to lie in an image of the very wrapping function it attacks. To avoid self-reference, the wrapping function has two branches. A branch-1 image consists of an injective tag of the preimage and the hash value. A branch-0 image also contains an obfuscated circuit that, on input a hash key, computes a branch-1 image of a preimage derived from that key by a PRF. The PRF key and the obfuscator's coins are derived from the preimage by a tag generator. Since the challenge image also contains the hash value of its preimage, the tag generator must stay pseudorandom given that hash value, which is the property defined next.

Definition B.5 (Leakage-resilient injective-tag generator). Let TG be a keyed function family with key generation TG.Kg , input length n , and output length $a + N$. For a key θ , write

$$\text{TG}(\theta, u) = \tau_\theta(u) \| T_\theta(u), \quad |\tau_\theta(u)| = a, \quad |T_\theta(u)| = N.$$

Then TG is an (ℓ, a) -leakage-resilient injective-tag generator (LRITG) if the following hold.

1. For every key $\theta \in [\text{TG.Kg}(1^\lambda)]$, the map $u \mapsto \tau_\theta(u)$ is injective.
2. For every non-uniform polynomial-size function $L : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ and every non-uniform PPT adversary $\mathcal{A}$, the advantage

$$\begin{aligned} \text{Adv}_{\text{TG}, L}^{(\ell, a)\text{-LRITG}}(\mathcal{A}) \stackrel{\text{def}}{=} & \left| \Pr[\mathcal{A}(\theta, \tau_\theta(u), L(u), T_\theta(u)) = 1] \right. \\ & \left. - \Pr[\mathcal{A}(\theta, \tau_\theta(u), L(u), U) = 1] \right| \end{aligned}$$

is negligible, where $\theta \leftarrow \$ \text{TG.Kg}(1^\lambda)$, $u \leftarrow \$ \{0, 1\}^n$, and $U \leftarrow \$ \{0, 1\}^N$.

We need non-uniform security because the proof fixes a hash key hk and applies the definition to the leakage function $u \mapsto H_{hk}(d||u)$. In Appendix B.4.1 we build an LRITG from an almost-always lossy trapdoor function and pairwise-independent hashing. The obfuscator may be taken to use a short random seed, using a standard PRG.

The two-branch wrapping function. We identify $(d, u) \in \{0, 1\} \times \{0, 1\}^n$ with the string $d||u$, so that the hash value of the preimage (d, u) is $H_{hk}(d||u)$. Let PRF be a punctured PRF with uniform k -bit keys, inputs in $\{0, 1\}^n \times \mathcal{K}_\lambda$, and outputs in $\{0, 1\}^n$. Let iO use r -bit coins with $r \geq \lambda$. Let TG be an (ℓ, a) -LRITG with $N = k + r$, and parse $T_\theta(u) = K_u||s_u$ with $|K_u| = k$ and $|s_u| = r$. For a key θ of TG and a PRF key K , define the circuit

$$\begin{aligned} & \frac{D_{\theta, K}(q, hk')}{u^* \leftarrow \text{PRF}_K(q, hk')} \\ & \tau^* || T^* \leftarrow \text{TG}(\theta, u^*) \\ & \text{Return } ((1, \tau^*), H_{hk'}(1||u^*)), \end{aligned} \tag{30}$$

padded to the size of the circuit $\tilde{D}$ in the proof of Proposition B.7. The wrapping function $\mathcal{G}_{\text{flip}} = \{g_\theta^{\text{flip}}\}$ has parameter θ and preimage domain $\mathcal{X} = \{0, 1\} \times \{0, 1\}^n$, and g_θ^{flip} is the algorithm in Figure 25. The encodings of the two kinds of image are canonical and disjoint.

Algorithm $g_\theta^{\text{flip}}((d, u), y)$ $\tau T \leftarrow \text{TG}(\theta, u)$ If $d = 1$: return $((1, \tau), y)$ Parse $T = K s$ $\Lambda \leftarrow iO(D_{\theta, K}; s)$ Return $((0, \Lambda, \tau), y)$

Figure 25: The two-branch wrapping function. A branch-0 image contains a circuit, and a branch-1 image is terminal.

The setup sampler and the preimage creator are

$$\begin{aligned} \text{Param}_{\text{flip}}(1^\lambda) : & \quad \theta \leftarrow \$ \text{TG.Kg}(1^\lambda); \quad u_{\text{ch}} \leftarrow \$ \{0, 1\}^n; \quad \text{return } (\theta, u_{\text{ch}}), \\ P_{\text{flip}}(hk, \theta, u_{\text{ch}}, w') : & \quad \text{return } ((0, u_{\text{ch}}), \varepsilon, \varepsilon). \end{aligned}$$

So the preimage creator ignores the hash key, and the challenge preimage is fixed by the hidden setup, as in FO. Write $\text{Wrap}_{\text{flip}} = (\mathcal{G}_{\text{flip}}, \text{Param}_{\text{flip}}, P_{\text{flip}})$.

Lemma B.6. $\mathcal{G}_{\text{flip}}$ is 1-PI, and its preimages, parameters, and images have polynomial length.

Proof. The consistency predicate of g_θ^{flip} rejects malformed inputs and otherwise checks $g_\theta^{\text{flip}}(x, y) = z$, which is efficient because g_θ^{flip} is deterministic and efficient. If an image z is consistent with $x = (d, u)$ and with $x' = (d', u')$, then its branch encoding gives $d = d'$ and its tag gives $\tau_\theta(u) = \tau_\theta(u')$, so $u = u'$ by injectivity of the tag. The last component of z is y , so at most one y is consistent with given x and z . The length bounds follow from the polynomial running times of TG, PRF, and iO . $\square$

To apply Proposition B.4, let $\text{Next}(1^\lambda, \text{hk}, \theta, z_{\text{ch}})$ parse $z_{\text{ch}} = ((0, \Lambda_{\text{ch}}, \tau_{\text{ch}}), y_{\text{ch}})$ and return $\Lambda_{\text{ch}}(0^n, \text{hk})$, or $\perp$ if parsing fails. Let $\text{Check}(\text{hk}, \theta, u_{\text{ch}}, z_{\text{ch}})$ return 1 exactly when z_{ch} is the genuine challenge image, that is, when

$$z_{\text{ch}} = g_{\theta}^{\text{flip}}((0, u_{\text{ch}}), H_{\text{hk}}(0 \| u_{\text{ch}})).$$

Proposition B.7 (Challenge-image counterexample). *Assume that TG is an (ℓ, a) -LRITG, PRF is a secure punctured PRF, and iO is an indistinguishability obfuscator. Then Hash does not satisfy 1-DINJEXT2[1] for $\text{Wrap}_{\text{flip}}$, and it does not satisfy 1-INJEXT2[1] for $\text{Wrap}_{\text{flip}}$ either.*

Below we check the two conditions of Proposition B.4. The first holds because the challenge image is a branch-0 image whose circuit outputs a valid branch-1 image with a PRF-derived preimage. For the second, the leakage resilience of TG is used twice. First, it replaces the PRF key and the obfuscator's coins in the challenge image by uniform strings, even given the tag and the hash value of the challenge preimage. Then, after the circuit has been replaced by one with the terminal image hardwired, and that image's preimage has been made uniform by puncturing the PRF, it turns any simulator that finds the terminal preimage into a distinguisher. The same proof shows that Hash does not satisfy 1-DINJEXT2[1] against QPT simulators if the three assumptions hold against non-uniform QPT adversaries. The adversary in that statement is classical.

Proof of Proposition B.7. By Lemma B.6, $\mathcal{G}_{\text{flip}}$ is 1-PI. We show that Next and Check satisfy the two conditions of Proposition B.4, which then gives the 1-DINJEXT2[1] statement. The 1-INJEXT2[1] statement follows from Proposition 5.8. Write C_{θ} for the consistency predicate of g_{θ}^{flip} .

The first condition. The challenge oracle uses the preimage $x_{\text{ch}} = (0, u_{\text{ch}})$. Write $T_{\theta}(u_{\text{ch}}) = K_{\text{ch}} \| s_{\text{ch}}$. The challenge image is

$$z_{\text{ch}} = ((0, \Lambda_{\text{ch}}, \tau_{\theta}(u_{\text{ch}})), H_{\text{hk}}(0 \| u_{\text{ch}})), \quad \Lambda_{\text{ch}} = iO(D_{\theta, K_{\text{ch}}}; s_{\text{ch}}).$$

Let $u^* = \text{PRF}_{K_{\text{ch}}}(0^n, \text{hk})$. By correctness of iO ,

$$\begin{aligned} \text{Next}(1^\lambda, \text{hk}, \theta, z_{\text{ch}}) &= \Lambda_{\text{ch}}(0^n, \text{hk}) \\ &= ((1, \tau_{\theta}(u^*)), H_{\text{hk}}(1 \| u^*)) \\ &= g_{\theta}^{\text{flip}}((1, u^*), H_{\text{hk}}(1 \| u^*)). \end{aligned}$$

So the image computed by Next is valid, and its preimage $(1, u^*)$ differs from $(0, u_{\text{ch}})$. Also, Check accepts the genuine challenge image. The first condition therefore holds with probability 1.

The second condition. Fix a non-uniform PPT algorithm $\mathcal{A}$ as in the second condition, and let $(\widehat{z}_{\text{ch}}, \widehat{x})$ be its output. Its challenge oracle returns z_{ch} on the first query and $\perp$ afterwards. Since Check accepts exactly the genuine challenge image, the event of the second condition is the event Good that

$$\widehat{z}_{\text{ch}} = z_{\text{ch}} \quad \text{and} \quad C_{\theta}(\widehat{x}, H_{\text{hk}}(\widehat{x}), \text{Next}(1^\lambda, \text{hk}, \theta, z_{\text{ch}})) = 1.$$

We bound $\Pr[\text{Good}]$ through a sequence of games. In each game the challenge oracle returns that game's z_{ch} on the first query and $\perp$ afterwards, and the game outputs 1 exactly when Good occurs.

Let $p_i = \Pr[G_i \Rightarrow 1]$. Every reduction below knows θ , hk , and z_{ch} , so it can check whether Good occurs.

Game G_0 is the experiment of the second condition, so $p_0 = \Pr[\text{Good}]$.

Game G_1 is G_0 with $T_\theta(u_{\text{ch}}) = K \| s_{\text{ch}}$ replaced by a uniform $K \| s \in \{0, 1\}^k \times \{0, 1\}^r$, so that

$$z_{\text{ch}} = ((0, iO(D_{\theta, K}; s), \tau_\theta(u_{\text{ch}})), H_{\text{hk}}(0 \| u_{\text{ch}})).$$

Fix, for each λ , a hash key hk_0 that maximizes $|\Pr[G_0 \Rightarrow 1 \mid \text{hk}] - \Pr[G_1 \Rightarrow 1 \mid \text{hk}]|$, and let $L_0(u) = H_{\text{hk}_0}(0 \| u)$. The distinguisher $\mathcal{D}_0$ has hk_0 as advice. On input (θ, τ, y, T) from the LRITG experiment for L_0 , it parses $T = K \| s$, sets $z_{\text{ch}} = ((0, iO(D_{\theta, K}; s), \tau), y)$, runs $\mathcal{A}$ with hash key hk_0 , and outputs 1 exactly when Good occurs. With $T = T_\theta(u)$ this is G_0 with $\text{hk} = \text{hk}_0$, where u plays the role of u_{ch} , and with uniform T it is G_1 with $\text{hk} = \text{hk}_0$. So

$$|p_0 - p_1| \leq \text{Adv}_{\text{TG}, L_0}^{(\ell, a)\text{-LRITG}}(\mathcal{D}_0).$$

Game G_2 . Let $\xi = (0^n, \text{hk})$, and let

$$u^* = \text{PRF}_K(\xi), \quad z^* = ((1, \tau_\theta(u^*)), H_{\text{hk}}(1 \| u^*)), \quad K\{\xi\} = \text{PRF.Punc}(K, \xi).$$

Game G_2 is G_1 with $D_{\theta, K}$ replaced by the circuit $\tilde{D}$ that outputs z^* on input ξ and, on every other input (q, hk') , outputs $((1, \tau_\theta(v)), H_{\text{hk}'}(1 \| v))$ for $v = \text{PRF}_{K\{\xi\}}(q, \text{hk}')$. The two circuits compute the same function, at ξ by the definition of z^* and elsewhere by punctured-PRF correctness, and they have the same size. The coins s are uniform in G_1 . So a distinguisher that samples everything except the obfuscation gives

$$|p_1 - p_2| \leq \epsilon_{iO}.$$

Game G_3 is G_2 with u^* in z^* replaced by a uniform string in $\{0, 1\}^n$, independent of $K\{\xi\}$. The punctured-PRF adversary $\mathcal{D}_1$ samples θ , hk , u_{ch} , and s , chooses $\xi = (0^n, \text{hk})$, receives $K\{\xi\}$ and a value for u^* , builds z_{ch} , and runs $\mathcal{A}$. So

$$|p_2 - p_3| \leq \text{Adv}_{\text{PRF}}^{\text{PPRF}}(\mathcal{D}_1).$$

Bounding p_3 . In G_3 , the value u^* is uniform and z^* is the branch-1 image of $(1, u^*)$. Fix, for each λ , a hash key hk_1 that maximizes $\Pr[G_3 \Rightarrow 1 \mid \text{hk}]$, and let $L_1(u) = H_{\text{hk}_1}(1 \| u)$. The distinguisher $\mathcal{D}_2$ has hk_1 as advice. On input (θ, τ, y, T) from the LRITG experiment for L_1 , it sets $z^* = ((1, \tau), y)$ and simulates G_3 with hash key hk_1 . That is, it samples u_{ch} , a PRF key K , and coins s , punctures K at $\xi = (0^n, \text{hk}_1)$, forms $\tilde{D}$ with z^* hardwired, and runs $\mathcal{A}$. The simulation does not use T , and it is exactly G_3 with $\text{hk} = \text{hk}_1$, where the LRITG input u plays the role of u^* . If Good occurs and $\hat{x} = (1, \hat{u})$, then $\mathcal{D}_2$ computes $\hat{\tau} \| \hat{T} = \text{TG}(\theta, \hat{u})$ and outputs $[\hat{T} = T]$. Otherwise it outputs 0.

If Good occurs, then $\hat{x}$ is the preimage of the branch-1 image z^* , so $\hat{x} = (1, \hat{u})$ with $\tau_\theta(\hat{u}) = \tau_\theta(u^*)$, and tag injectivity gives $\hat{u} = u^*$. So with $T = T_\theta(u^*)$ the distinguisher outputs 1 whenever Good occurs. With uniform T it outputs 1 with probability at most 2^{-N} , because T is independent of everything else that it computes. Therefore

$$p_3 \leq \Pr[G_3 \Rightarrow 1 \mid \text{hk} = \text{hk}_1] \leq \text{Adv}_{\text{TG}, L_1}^{(\ell, a)\text{-LRITG}}(\mathcal{D}_2) + 2^{-N}.$$

Combining the games gives

$$\begin{aligned} \Pr[\text{Good}] &\leq \text{Adv}_{\text{TG}, L_0}^{(\ell, a)\text{-LRITG}}(\mathcal{D}_0) + \epsilon_{iO} + \text{Adv}_{\text{PRF}}^{\text{PPRF}}(\mathcal{D}_1) \\ &\quad + \text{Adv}_{\text{TG}, L_1}^{(\ell, a)\text{-LRITG}}(\mathcal{D}_2) + 2^{-N}, \end{aligned}$$

which is negligible because $N \geq r \geq \lambda$. This proves the second condition. $\square$

B.4.1 Leakage-Resilient Injective-Tag Generators from Lossy TDFs

We build the tag generator used in Proposition B.7 from an almost-always lossy trapdoor function and pairwise-independent hashing. Let

$$\text{LT} = (\text{IKg}, \text{LKg}, \text{Eval}, \text{Inv})$$

be an almost-always lossy trapdoor function (Definition 2.9) with domain $\{0, 1\}^n$, range $\{0, 1\}^a$, lossiness ℓ_{loss} , and failure probability β_{loss} . Let $\mathcal{H} = \{h_\sigma : \{0, 1\}^n \rightarrow \{0, 1\}^N\}_{\sigma \in \mathcal{S}}$ be a pairwise-independent hash function family. Define TG by

$$\begin{aligned} &\text{TG.Kg}(1^\lambda) \\ & \quad (ek, td) \leftarrow \text{IKg}(1^\lambda) \quad \text{TG}((ek, \sigma), u) = \text{Eval}(ek, u) \parallel h_\sigma(u). \\ & \quad \sigma \leftarrow \mathcal{S} \\ & \quad \text{Return } \theta = (ek, \sigma), \end{aligned}$$

So $\tau_\theta(u) = \text{Eval}(ek, u)$ and $T_\theta(u) = h_\sigma(u)$.

Theorem B.8. *Let β_{loss} and ϵ be negligible, and suppose that*

$$\ell_{\text{loss}} \geq \ell + N + 2\log(1/\epsilon).$$

Then TG is an (ℓ, a) -leakage-resilient injective-tag generator. The same holds against non-uniform QPT adversaries if the mode indistinguishability of LT does.

Proof. An injective-mode key makes $u \mapsto \text{Eval}(ek, u)$ injective, so the tag is injective. For leakage resilience, fix a non-uniform polynomial-size function $L : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ and a non-uniform PPT adversary $\mathcal{A}$. In the first experiment of Definition B.5, the adversary receives

$$((ek, \sigma), \text{Eval}(ek, u), L(u), h_\sigma(u))$$

for $(ek, td) \leftarrow \text{IKg}(1^\lambda)$, $\sigma \leftarrow \mathcal{S}$, and $u \leftarrow \{0, 1\}^n$. We change this experiment in three steps.

First, sample $ek \leftarrow \text{LKg}(1^\lambda)$ instead. A reduction that receives ek samples σ and u , forms the tuple, and runs $\mathcal{A}$, so this changes the output probability by at most a mode-distinguishing advantage.

Second, replace $h_\sigma(u)$ by a uniform $U \in \{0, 1\}^N$. Fix a lossy key ek whose image has at most $2^{n-\ell_{\text{loss}}}$ elements. A random lossy key has this property except with probability β_{loss} . The pair $V = (\text{Eval}(ek, u), L(u))$ is a function of the uniform u with at most $2^{n-\ell_{\text{loss}}+\ell}$ values. If N_v is the number of u with $V = v$, then

$$\text{cp}(u \mid V) = \sum_v \frac{N_v}{2^n} \cdot \frac{1}{N_v} = \frac{|\text{Im}(V)|}{2^n} \leq 2^{\ell-\ell_{\text{loss}}}.$$

A pairwise-independent family is two-universal, and σ is independent of (u, V) . So Lemma 2.16 gives

$$\text{SD}((V, \sigma, h_\sigma(u)), (V, \sigma, U)) \leq \frac{1}{2} \sqrt{2^{N+\ell-\ell_{\text{loss}}}} \leq \frac{\epsilon}{2},$$

using $\ell_{\text{loss}} \geq \ell + N + 2 \log(1/\epsilon)$. So this step changes the output probability by at most $\beta_{\text{loss}} + \epsilon/2$.

Third, sample ek in injective mode again. The same reduction, now using U in place of $h_\sigma(u)$, bounds this change by a second mode-distinguishing advantage. The result is the second experiment of Definition B.5. So the LRITG advantage is at most two mode-distinguishing advantages plus $\beta_{\text{loss}} + \epsilon/2$, which is negligible. For QPT adversaries the argument is the same, because the second step is statistical. $\square$

For the two-branch wrapping function, $N = k + r$, so the required lossiness is $\ell_{\text{loss}} \geq \ell + k + r + 2 \log(1/\epsilon)$.

C Proofs for the ROM and QROM Results

C.1 Proof of Proposition 7.3

Proof. Fix a wrapping triple $(\mathcal{G}, \text{Param}, P)$ with $\mathcal{G}$ Γ -PI, and fix $\mathcal{B} = (\mathcal{B}_{\text{run}}, \mathcal{B}_{\text{out}})$. In this proof, Γ denotes the value $\Gamma(\lambda, t_{\text{max}})$.

Removing the sampling oracle. Let $\mathcal{B}'$ run $\mathcal{B}_{\text{run}}$ and answer each call $O_{\text{samp}}(w)$ itself. If $w \notin \mathcal{W}$, it returns $\perp$. Otherwise it samples $x \leftarrow \mathcal{X}$ and $\rho \leftarrow \mathcal{R}$, queries $y \leftarrow G(x)$, and returns $g_\rho(x, y, w; \rho)$. It forwards every other query to its own oracles, and its output predicate is $\mathcal{B}_{\text{out}}$. Write $\text{Inv}_{\text{RO}}(\mathcal{B}')$ for the ideal inversion game of Figure 22 with $\mathcal{B}'$ in place of $\mathcal{B}$ and without O_{samp} . The sampling replies have the same distribution in the two games, so

$$\Pr[\text{Inv-1-DINJEXT2-samp}_{\text{RO}, \mathcal{G}}^{\mathcal{B}} \Rightarrow 1] = \Pr[\text{Inv}_{\text{RO}}(\mathcal{B}') \Rightarrow 1].$$

The adversary $\mathcal{B}'$ makes the same inversion and challenge queries as $\mathcal{B}$.

Preserving the condition on challenge queries. Let $\mathcal{B}'^-$ be $\mathcal{B}'$ with each challenge query answered locally by $\perp$. Set $\mathcal{A} = \mathcal{B}'^-$ if $q_{\text{chal}} = 0$ with probability 1 for every λ in the ideal inversion game for $\mathcal{B}$, and set $\mathcal{A} = \mathcal{B}'$ otherwise. Which case applies depends only on $\mathcal{B}$, so $\mathcal{A}$ is PPT. In the first case, the ideal executions with $\mathcal{B}'^-$ and $\mathcal{B}'$ agree until a challenge query in $\mathcal{W}'$, which occurs with probability 0. Queries outside $\mathcal{W}'$ receive $\perp$ in both executions. Thus, in either case,

$$\Pr[\text{Inv-1-DINJEXT2-samp}_{\text{RO}, \mathcal{G}}^{\mathcal{B}} \Rightarrow 1] = \Pr[\text{Inv}_{\text{RO}}(\mathcal{A}) \Rightarrow 1]. \quad (31)$$

The run stage of $\mathcal{A}$ makes at most q_{inv} inversion queries, each of length at most t_{max} , and its output predicate is $\mathcal{B}_{\text{out}}$.

The simulator. On input $(1^\lambda, p)$, the simulator $\mathcal{R}$ runs $\mathcal{A}_{\text{run}}(1^\lambda, p)$ and keeps an empty table T and an empty list L_{chal} . It answers each query x to G by querying its own oracle, recording $T[x] = G(x)$, and returning this value. It forwards challenge queries to its own oracle O_{chal} and records the challenge image, if one is returned, in L_{chal} . To answer an inversion query z , it returns $\perp$ if $z \in L_{\text{chal}}$. Otherwise it searches T for an x with $C_p(x, T[x], z) = 1$. If there is none, it returns $\perp$. If there is one, it is unique by x -injectivity. The simulator returns this x unless $C_p(x, T[x], z^*) = 1$ for some $z^* \in L_{\text{chal}}$, in which case it returns $\perp$. Finally, it outputs the output v of $\mathcal{A}_{\text{run}}$.

By x -injectivity, the last test rejects exactly when x is the preimage of a registered challenge image, as required by O_{inv} . All searches are over the polynomial-size table T , so $\mathcal{R}$ is PPT.

Analysis. Compare $\text{Inv}_{\text{RO}}(\mathcal{A})$ with the simulation game of Figure 22 using $\mathcal{R}$. Run them on the same G and the same coins for the setup sampler, the preimage creator, and $\mathcal{A}_{\text{run}}$. They use the same output predicate, and differ only in how inversion queries are answered, so they proceed identically until the first query on which the two answers differ. Writing bad for the event that some query receives different answers, Equation (31) gives

$$\text{Adv}_{\text{RO}, G, \text{Param}, P}^{1\text{-DINJEXT2-samp}}(\mathcal{B}, \mathcal{R}) \leq \Pr[\text{bad}].$$

We first identify when a single query can receive different answers. Consider a query z and suppose every earlier query received the same answer in both games. If $z \in L_{\text{chal}}$, both oracles return $\perp$. Otherwise, let x be the unique element of $\mathcal{X}$ with $C_p(x, G(x), z) = 1$, if there is one. If there is none, then O_{inv} returns $\perp$, and so does the simulator. If x exists and is the x -preimage of a registered challenge image $z^* \neq z$, both return $\perp$. If x exists and lies in $\text{dom}(T)$, both return x . So the two answers differ only when x exists, is not the preimage of a registered challenge image, and is not in $\text{dom}(T)$; in that case O_{inv} returns x and the simulator returns $\perp$.

It remains to bound the probability that some query falls in this last case. For $x \in \mathcal{X}$ and a string z put

$$Y_{x,z} \stackrel{\text{def}}{=} \{y \in \{0,1\}^\ell : C_p(x, y, z) = 1\},$$

so that $|Y_{x,z}| \leq \Gamma$ for $|z| \leq t_{\text{max}}$ by y -ambiguity, since $\Gamma(\lambda, t)$ is nondecreasing in t , and for a fixed z at most one x has $Y_{x,z} \neq \emptyset$ by x -injectivity. The predicate C_p has no oracle access, so the sets $Y_{x,z}$ are determined by p, x , and z without further evaluation of G .

Run the simulation game with G sampled lazily. During the run stage, this execution evaluates G only at the points of $\text{dom}(T)$, including the queries used to answer sampling calls, and at the preimage x^* of the registered challenge image once the challenge query is made. Indeed, the honest algorithms have no oracle access, and the simulator answers inversion queries using only T and C_p . Let $z_1, \dots, z_{q_{\text{inv}}}$ denote the inversion queries, omitting indices for queries that are not made. For each j , let $\hat{x}_j$ be the unique $x \in \mathcal{X}$ with $Y_{x,z_j} \neq \emptyset$, if there is one, which is determined by p and z_j . Let bad_j be the event that all of the following hold at the time of the j th query.

- $z_j \notin L_{\text{chal}}$ and $\hat{x}_j$ exists.
- $\hat{x}_j \notin \text{dom}(T)$, and $\hat{x}_j \neq x^*$ if a challenge image has been registered.
- $G(\hat{x}_j) \in Y_{\hat{x}_j, z_j}$.

The last condition says exactly that $\hat{x}_j$ is the element x in the case analysis above. So, by that case analysis, bad occurs if and only if some bad_j does.

Fix j and condition on the execution up to the j th query. This fixes z_j , and hence $\hat{x}_j$ and $Y_{\hat{x}_j, z_j}$. If the first two conditions of bad_j fail, then bad_j does not occur. If they hold, then G has not yet been evaluated at $\hat{x}_j$. The value $G(\hat{x}_j)$ is therefore uniform on $\{0,1\}^\ell$ and independent of the conditioning. Thus the conditional probability of bad_j is at most $|Y_{\hat{x}_j, z_j}|/2^\ell \leq \Gamma/2^\ell$. Averaging gives $\Pr[\text{bad}_j] \leq \Gamma/2^\ell$. A union bound over the at most q_{inv} queries gives

$$\text{Adv}_{\text{RO}, G, \text{Param}, P}^{1\text{-DINJEXT2-samp}}(\mathcal{B}, \mathcal{R}) \leq q_{\text{inv}} \cdot \frac{\Gamma}{2^\ell}.$$

Finally, when the condition on challenge queries applies to $\mathcal{B}$, the algorithm $\mathcal{A}$ never queries the challenge oracle, so neither does $\mathcal{R}$. This proves the proposition. $\square$

C.2 Proof of Theorem 7.4

We prove Theorem 7.4 by replacing the simulator's replies with ideal inversion replies one at a time. Write $\Gamma = \Gamma_{SE}(\lambda, t_{\max})$ and $\ell = \ell_K$. The main step is Lemma C.1, which compares the two replies together with the remaining quantum state.

Quantum notation. We use registers for finite-dimensional quantum systems, $|\psi\rangle$ for a pure state, and τ for a density operator. We write $\|A\|$ for the operator norm and $\|A\|_1 = \text{tr} \sqrt{A^\dagger A}$ for the trace norm. The trace distance of two states τ, τ' is $\frac{1}{2}\|\tau - \tau'\|_1$. It bounds the difference in the acceptance probabilities of any subsequent test and cannot increase under a common quantum channel, meaning a completely positive, trace-preserving map. For classical states, it equals the statistical distance defined in the preliminaries. Let $\tau = \sum_c \text{Pr}[c] |c\rangle\langle c| \otimes \tau_c$ be a classical-quantum state, and consider two channels that leave the classical register c unchanged and act on the rest of the state depending on c . Then the trace distance of their outputs on τ is the average, weighted by $\text{Pr}[c]$, of the trace distances of their outputs on the states τ_c .

The compressed oracle. We use the formulation by Don, Fehr, Majenz, and Schaffner (DFMS) [41] of Zhandry's compressed oracle [74]. Let $\mathcal{U}$ be a finite, efficiently ordered set, and let $\mathcal{Y} = \{0, 1\}^\ell$. A random function $\mathcal{U} \rightarrow \mathcal{Y}$ is represented by a database register $D = (D_x)_{x \in \mathcal{U}}$. Each cell D_x has orthonormal basis $\{|y\rangle : y \in \mathcal{Y}\} \cup \{|\perp\rangle\}$ and starts in $|\perp\rangle$. The symbol $\perp$ marks an empty cell; it is not a value of the function. Let

$$|u\rangle \stackrel{\text{def}}{=} 2^{-\ell/2} \sum_{y \in \mathcal{Y}} |y\rangle,$$

and let Φ be the unitary on one cell that exchanges $|\perp\rangle$ and $|u\rangle$ and fixes every vector orthogonal to both, so that Φ is its own inverse. A query with input register Q and answer register Y is answered by the unitary

$$O \stackrel{\text{def}}{=} \sum_{x \in \mathcal{U}} |x\rangle\langle x|_Q \otimes \Phi_{D_x} \text{CNOT}_{D_x Y} \Phi_{D_x},$$

where $\text{CNOT}_{D_x Y}$ maps $|y\rangle_{D_x} |a\rangle_Y$ to $|y\rangle_{D_x} |a \oplus y\rangle_Y$ and acts as the identity when D_x is in $|\perp\rangle$. We call $\Phi_{D_x} \text{CNOT}_{D_x Y} \Phi_{D_x}$ the block of O at x .

Let $\mathcal{V}_x$ be the subspace of the cell D_x orthogonal to $|u\rangle$, and put $\mathcal{V}_D \stackrel{\text{def}}{=} \bigotimes_{x \in \mathcal{U}} \mathcal{V}_x$. A joint state is *supported on* $\mathcal{V}_D$ if its support lies in $\mathcal{V}_D$ tensored with the spaces of all other registers. The map Φ sends the span of $\{|y\rangle : y \in \mathcal{Y}\}$ onto $\mathcal{V}_x$, so $\text{Comp} \stackrel{\text{def}}{=} \bigotimes_{x \in \mathcal{U}} \Phi_{D_x}$ sends $\mathcal{V}_D$ onto the span of the product states $\bigotimes_x |f(x)\rangle_{D_x}$ for functions $f : \mathcal{U} \rightarrow \mathcal{Y}$. We identify this span with a register that holds a function table f and call it the *table representation*. On it, $\sum_x |x\rangle\langle x|_Q \otimes \text{CNOT}_{D_x Y}$ is the standard oracle O^{st} , which maps $|x\rangle |a\rangle |f\rangle$ to $|x\rangle |a \oplus f(x)\rangle |f\rangle$. Since Φ is its own inverse, we have $O = \text{Comp} (\sum_x |x\rangle\langle x|_Q \otimes \text{CNOT}_{D_x Y}) \text{Comp}$, and hence O acts on states supported on $\mathcal{V}_D$ as $\text{Comp} O^{\text{st}} \text{Comp}$. The initial database $|\perp\rangle^{\otimes \mathcal{U}}$ is Comp applied to the uniform superposition of all function tables. So, as long as the database is acted on only by O , the compressed oracle is unitarily equivalent, via Comp , to the purified random oracle, and it answers queries exactly as a uniformly random function does [74, 41]. In the table representation, O^{st} acts on the table only as a control in its computational basis.

The extraction measurement. For a relation $R \subseteq \mathcal{U} \times \mathcal{Y}$, let $\Gamma_R \stackrel{\text{def}}{=} \max_x |\{y : (x, y) \in R\}|$. The relation is *x-unique* if at most one x has $(x, y) \in R$ for some y , and we then write x_R for that input when it exists. The extraction measurement M^R of DFMS returns the least x whose cell D_x holds some y with $(x, y) \in R$, and returns $\perp$ if there is none. For an x -unique R only the cell D_{x_R} matters, and M^R is the projective measurement $\{\Pi, I - \Pi\}$ on that cell, with outcomes x_R and $\perp$, where

$$\Pi \stackrel{\text{def}}{=} \sum_{y:(x_R, y) \in R} |y\rangle\langle y|_{D_{x_R}}.$$

If R is empty, M^R returns $\perp$ and does nothing. Since $\Pi|\perp\rangle = 0$, a cell in the state $|\perp\rangle$ always gives the outcome $\perp$ and is left unchanged. Using the sparse representation of the database, which stores only its nonempty cells, M^R can be performed in quantum polynomial time whenever membership in R can be decided in polynomial time, by testing every nonempty cell [41]. The same representation lets each query O be answered in polynomial time.

Ideal inversion. Let R be x -unique. The *ideal inversion operation* $\widehat{M}^R$ returns $\perp$ if R is empty. Otherwise it applies the block of O at x_R with a fresh answer register, tests whether the answer y satisfies $(x_R, y) \in R$, and copies x_R or $\perp$, according to the test, to an output register. It then reverses the test and applies the block of O at x_R again, which restores the answer register, and measures the output register. In the table representation this is the projective measurement of the table entry $f(x_R)$ onto the values y with $(x_R, y) \in R$. Finding x_R may require exhaustive search, which is acceptable because the operation is used only to describe ideal inversion in the hybrid games below.

Lemma C.1 (Replacing one extraction reply). *Let $R \subseteq \mathcal{U} \times \mathcal{Y}$ be x -unique with $\Gamma_R \leq \Gamma$.*

1. *The initial database is supported on $\mathcal{V}_D$. The query O , each block of O , ideal inversion operations $\widehat{M}^{R'}$ for x -unique relations R' , operations that do not act on D , and unitaries that apply one of these operations controlled on a basis state of another register all preserve this support condition, also after conditioning on classical outcomes.*
2. *For every joint state τ supported on $\mathcal{V}_D$,*

$$\frac{1}{2} \|M^R(\tau) - \widehat{M}^R(\tau)\|_1 \leq 2\sqrt{\frac{\Gamma}{2^\ell}}.$$

Both output states include the classical answer, the database, and all remaining registers of τ .

Proof. Item 1. Initially every cell is in $|\perp\rangle \in \mathcal{V}_x$. On $\mathcal{V}_D$ the query O acts as $\text{Comp } O^{\text{st}} \text{ Comp}$, and the block of O at x acts as $\text{Comp CNOT}_{D_x Y} \text{ Comp}$. Both O^{st} and $\text{CNOT}_{D_x Y}$ preserve the span of the function tables, and Comp maps that span onto $\mathcal{V}_D$, so the query and its blocks preserve the support condition. Operations that do not act on D , including measurements of other registers, preserve the condition trivially. An ideal inversion operation consists of two blocks of O , a test and a copy that do not act on D , and a measurement of its output register, so it preserves the condition as well. A unitary that applies one of these operations controlled on a basis state of another register acts separately on each basis state of that register, so it also preserves the condition. Finally, conditioning on a classical outcome keeps the support inside the same subspace.

Item 2. If R is empty, both operations return $\perp$ and the distance is zero. Otherwise let

$$T_R \stackrel{\text{def}}{=} \{y : (x_R, y) \in R\}, \quad r \stackrel{\text{def}}{=} |T_R| \leq \Gamma,$$

and let Π be the projector onto the span of $\{|y\rangle : y \in T_R\}$ in the cell D_{x_R} . The measurement M^R is $\{\Pi, I - \Pi\}$, with respective answers x_R and $\perp$. We show that on states supported on $\mathcal{V}_D$, the ideal inversion operation $\widehat{M}^R$ is the measurement $\{\widehat{\Pi}, I - \widehat{\Pi}\}$ on the same cell, where

$$\widehat{\Pi} = \Phi \Pi \Phi.$$

Let the cell be in a state $\phi \in \mathcal{V}_{x_R}$, and write $\Phi\phi = \sum_y \alpha_y |y\rangle$, which is possible because Φ maps $\mathcal{V}_{x_R}$ into the span of the $|y\rangle$. The first block of O gives $\sum_y \alpha_y \Phi |y\rangle |y\rangle_Y$, and the test and the copy give $\sum_y \alpha_y \Phi |y\rangle |y\rangle_Y |b_y\rangle$, where $b_y = x_R$ if $y \in T_R$ and $b_y = \perp$ otherwise. The second block of O first applies Φ , which turns $\Phi |y\rangle$ back into $|y\rangle$, then clears Y , and then applies Φ again. The result is $\Phi \Pi \Phi \phi \otimes |x_R\rangle + \Phi(I - \Pi)\Phi\phi \otimes |\perp\rangle$, with the answer register back in $|0\rangle$. By linearity, the same holds for joint states supported on $\mathcal{V}_D$, which proves the claim.

We bound $\|\Pi - \widehat{\Pi}\|$. The projectors Π and $\Phi \Pi \Phi$ share the $(r-1)$ -dimensional subspace of vectors supported on T_R whose coefficients sum to zero, because Φ fixes these vectors. Their remaining unit vectors are

$$|s\rangle = r^{-1/2} \sum_{y \in T_R} |y\rangle \quad \text{and} \quad \Phi |s\rangle, \quad \langle s | \Phi | s \rangle = 1 - r/2^\ell,$$

and both are orthogonal to the shared subspace. Consequently,

$$\|\Pi - \widehat{\Pi}\| = \sqrt{1 - (1 - r/2^\ell)^2} \leq \sqrt{2r/2^\ell}.$$

The same bound appears in DFMS [41].

Purify the input state, retaining all reference registers. Before the output register is measured, the two operations have the isometries

$$V_\Pi |\psi\rangle = \Pi |\psi\rangle |x_R\rangle + (I - \Pi) |\psi\rangle |\perp\rangle$$

and $V_{\widehat{\Pi}}$, defined in the same way with $\widehat{\Pi}$. For every normalized $|\psi\rangle$,

$$\|(V_\Pi - V_{\widehat{\Pi}}) |\psi\rangle\|^2 = 2\|(\Pi - \widehat{\Pi}) |\psi\rangle\|^2 \leq \frac{4r}{2^\ell},$$

so $\|(V_\Pi - V_{\widehat{\Pi}}) |\psi\rangle\| \leq 2\sqrt{\Gamma/2^\ell}$. The trace distance of two pure states is at most their Euclidean distance, and measuring the output register and discarding purification registers cannot increase trace distance. This proves item 2 also for mixed input states. $\square$

Item 1 applies only before the first extraction measurement, because M^R need not preserve $\mathcal{V}_D$. This suffices, because item 2 compares the complete states at that point, and no common continuation, including further oracle queries and extraction measurements, can increase their trace distance.

Proof of Theorem 7.4. Fix $\mathcal{B} = (\mathcal{B}_{\text{run}}, \mathcal{B}_{\text{out}})$. Construct $\mathcal{B}'$ and $\mathcal{A}$ for the triple Wrap_{FO} as in Appendix C.1, using the QROM ideal inversion game for the condition on challenge queries. Both modifications use only classical sampling and classical challenge queries, so they also give QPT

algorithms in the QROM. Write $\text{Inv}_{\text{QRO}}(\mathcal{A})$ for the ideal inversion game of Figure 23 with $\mathcal{A}$ and without the sampling oracle. The same argument gives

$$\Pr[\text{Inv-1-DINJEXT2-samp}_{\text{QRO,WrapFO}}^{\mathcal{B}} \Rightarrow 1] = \Pr[\text{Inv}_{\text{QRO}}(\mathcal{A}) \Rightarrow 1]. \quad (32)$$

The simulator. Let $\mathcal{U} = \{0,1\}^n$. The simulator uses the interface of Definition 7.2, with the compressed database D as its private register S . The subroutine $\mathcal{R}.\text{Init}(1^\lambda)$ prepares the empty database, representing $|\perp\rangle^{\otimes \mathcal{U}}$. The subroutine $\mathcal{R}.\text{RO}$ applies the query O above. It has the form required in Definition 7.2 and makes no oracle calls.

The subroutine $\mathcal{R}.\text{Sim}$ runs $\mathcal{A}_{\text{run}}(1^\lambda, p)$ and starts with an empty list L_{chal} . It forwards queries to $\tilde{G}$ and O_{chal} to its own oracles and records the challenge image, if one is returned, in L_{chal} . It answers an inversion query z with $\perp$ if $z \in L_{\text{chal}}$. Otherwise it applies $M^{R_{p,z}}$ to D for the relation

$$R_{p,z} \stackrel{\text{def}}{=} \{(x, y) \in \mathcal{U} \times \mathcal{Y} : C_p(x, y, z) = 1\},$$

and returns the outcome. It outputs the output v of $\mathcal{A}_{\text{run}}$. The predicate C_p has no oracle access, so the simulator can evaluate it coherently without querying G . Membership in $R_{p,z}$ takes one evaluation of C_p . The relation is x -unique by x -injectivity, and $\Gamma_{R_{p,z}} \leq \Gamma$ by y -ambiguity whenever $|z| \leq t_{\text{max}}$. The database has a sparse representation, so all three subroutines are QPT.

Analysis. For an image z , let $\text{cand}(z)$ be the unique $x \in X$ with $C_p(x, y, z) = 1$ for some y , if there is one. It is unique by x -injectivity and determined by p and z , and we call it the candidate of z .

The cell at the challenge preimage. We show that the cell D_{x^*} stays in the state $|\perp\rangle$, in tensor product with all other registers, throughout the simulation game and the hybrids below. Let x^* and y^* be as in Figure 23, and let z^* be the challenge image once it exists. The preimage creator outputs x^* , so x^* is the preimage of z^* . No algorithm applies the block of O at x^* . Indeed, the setup sampler, the preimage creator, the wrapping function, and the consistency predicate have no oracle access, and the game answers every query at x^* with y^* itself, including the challenge oracle's evaluation $\tilde{G}(x^*) = y^*$. The ideal replies in the hybrids below never act on the cell at x^* either. A measurement M^R with $x_R \neq x^*$ acts only on the cell D_{x_R} , and a measurement M^R with $x_R = x^*$ applied to a cell in the state $|\perp\rangle$ returns $\perp$ with probability 1 and leaves the state unchanged, since $\Pi|\perp\rangle = 0$. By induction over the operations of the game, the cell D_{x^*} therefore stays in $|\perp\rangle$, and every measurement M^R with $x_R = x^*$ returns $\perp$ and leaves the state unchanged.

The ideal reply. On a basis state $f : \mathcal{U} \rightarrow \mathcal{Y}$ of D in the table representation, the oracle $\tilde{G}$ evaluates the function that equals f except that $\tilde{G}(x^*) = y^*$. Define the *ideal reply* to an inversion query z as follows.

- If $z \in L_{\text{chal}}$, or z has no candidate, it is $\perp$.
- If $z \notin L_{\text{chal}}$, and $x_z = \text{cand}(z)$ exists with $x_z \neq x^*$, it is the operation $\hat{M}^{R_{p,z}}$ on D .
- If $z \notin L_{\text{chal}}$ and $\text{cand}(z) = x^*$, it is x^* when $C_p(x^*, y^*, z) = 1$ and I_{chal} contains no pair (x^*, z') with $z' \neq z$, and it is $\perp$ otherwise. This reply does not act on D .

Write $O_{\text{inv}}^{\tilde{G}}$ for the ideal inverter of Figure 22, with $\tilde{G}$ in place of G . We claim that, on each basis state of the table representation of D , the ideal reply to z returns $O_{\text{inv}}^{\tilde{G}}(z)$ with probability 1 and leaves the state unchanged. The first case matches the first two tests of the ideal inverter. In the second case, I_{chal} contains at most the pair (x^*, z^*) , since the preimage creator outputs x^* , and $x_z \neq x^*$,

so the ideal inverter returns x_z exactly when $C_p(x_z, \tilde{G}(x_z), z) = 1$. Here $\tilde{G}(x_z) = f(x_z)$, so this is the condition $(x_z, f(x_z)) \in R_{p,z}$ that $\hat{M}^{R_{p,z}}$ tests, and on a basis state of the table this test has a certain outcome and does not change the state. The third case matches the ideal inverter because $\tilde{G}(x^*) = y^*$.

Hybrids. For $i \in \{0, \dots, q_{\text{inv}}\}$, let Hyb_i be the simulation game of Figure 23 with $\mathcal{R}$ above, except that the first i inversion queries of $\mathcal{A}_{\text{run}}$ receive the ideal reply. Then Hyb_0 is the simulation game.

We show that $\Pr[\text{Hyb}_{q_{\text{inv}}} \Rightarrow 1]$ equals the probability that $\text{Inv}_{\text{QRO}}(\mathcal{A})$ outputs 1. In $\text{Hyb}_{q_{\text{inv}}}$ no measurement M^R is applied, so the database D is acted on only by blocks of O and by ideal inversion operations, all of which act on f only as a control or as a measurement in its computational basis. We may therefore measure f at the start without changing the distribution of the game. Then f and y^* are uniform and independent of each other and of the coins of the setup sampler. For each fixing of these coins, and hence of x^* , the function $\tilde{G}$ is therefore uniform, so $\tilde{G}$ is uniform and independent of (p, aux) . In $\text{Hyb}_{q_{\text{inv}}}$ every algorithm uses the oracle $\tilde{G}$ in the way the ideal inversion game uses G . The setup sampler has no oracle access, and the challenge oracle forms the challenge image from $\tilde{G}(x^*) = y^*$. The replies to inversion queries are $O_{\text{inv}}^{\tilde{G}}$ by the claim above, and the adversary's run stage and output predicate query $\tilde{G}$. So $\text{Hyb}_{q_{\text{inv}}}$ and the ideal inversion game output 1 with the same probability.

Adjacent hybrids. Fix $i < q_{\text{inv}}$. The games Hyb_i and Hyb_{i+1} agree until the $(i+1)$ st inversion query, and afterwards they apply the same continuation. If that query is not made, the two games are identical. Otherwise let τ be the joint state just before the query is answered. It is a classical-quantum state whose classical part c includes the query z , the lists L_{chal} and I_{chal} , and the values x^* , y^* , and p . No measurement M^R has been applied so far, and for each query input $\tilde{G}$ either applies a block of O or adds a fixed value y^* to the answer register. So, by item 1 of Lemma C.1, each τ_c is supported on $\mathcal{V}_D$. Let δ_c be the trace distance between the states after the two replies to z on input τ_c . Then

$$|\Pr[\text{Hyb}_i \Rightarrow 1] - \Pr[\text{Hyb}_{i+1} \Rightarrow 1]| \leq \sum_c \Pr[c] \delta_c.$$

We bound δ_c in three cases.

- $z \in L_{\text{chal}}$, or z has no candidate, or $\text{cand}(z) \neq x^*$. If $z \in L_{\text{chal}}$, both games return $\perp$. If z has no candidate, then $R_{p,z}$ is empty and both games return $\perp$. Otherwise the simulator applies $M^{R_{p,z}}$ and the ideal reply is $\hat{M}^{R_{p,z}}$, and item 2 of Lemma C.1 gives $\delta_c \leq 2\sqrt{\Gamma/2^\ell}$.
- $\text{cand}(z) = x^*$, $z \notin L_{\text{chal}}$, and the challenge image has been registered. The simulator returns $\perp$ and leaves the state unchanged, as shown above, and the ideal reply is $\perp$, because $(x^*, z^*) \in I_{\text{chal}}$ and $z \neq z^*$. So $\delta_c = 0$.
- $\text{cand}(z) = x^*$, $z \notin L_{\text{chal}}$, and no challenge image has been registered. Here we use only $\delta_c \leq 1$.

Let early_{i+1} be the event of the third case. Then

$$|\Pr[\text{Hyb}_i \Rightarrow 1] - \Pr[\text{Hyb}_{i+1} \Rightarrow 1]| \leq 2\sqrt{\frac{\Gamma}{2^\ell}} + \Pr[\text{early}_{i+1}],$$

where the probability is taken in the part of the games before the reply.

Early queries at the challenge preimage. It remains to show that $\Pr[\text{early}_{i+1}] \leq 1/|\mathcal{X}|$. Consider the run stage up to the earlier of its $(i+1)$ st inversion query and its first challenge query with an

input in $\mathcal{W}'$. In this part of the games every inversion query receives the ideal reply, the lists L_{chal} and I_{chal} are empty, and no measurement M^R is applied. So, as in the argument for $\text{Hyb}_{q_{\text{inv}}}$, we may measure f at the start, and $\tilde{G}$ is then a uniform function independent of (p, aux) . Each ideal reply in this part is $O_{\text{inv}}^{\tilde{G}}(z)$ with empty lists, which is determined by p , z , and $\tilde{G}$. So the run stage's behavior in this part is determined by p , by $\tilde{G}$, and by its own randomness. The setup sampler samples x^* uniformly from $\mathcal{X} = \text{F.Dom}$ and independently of p . Since $\tilde{G}$ is independent of (p, aux) , the value x^* is uniform and independent of $(p, \tilde{G})$, and hence of everything that determines the run stage in this part. The event early_{i+1} requires the $(i+1)$ st inversion query z to be made in this part and to have $\text{cand}(z) = x^*$. Since $\text{cand}(z)$ is determined by p and z , this has probability at most $1/|\mathcal{X}|$.

Conclusion. Summing over $i < q_{\text{inv}}$ gives

$$|\Pr[\text{Hyb}_0 \Rightarrow 1] - \Pr[\text{Hyb}_{q_{\text{inv}}} \Rightarrow 1]| \leq q_{\text{inv}} \left(2\sqrt{\frac{\Gamma}{2^\ell}} + \frac{1}{|\mathcal{X}|} \right).$$

By Equation (32), this bounds $\text{Adv}_{\text{QRO}, \text{Wrap}_{\text{FO}}}^{1\text{-DINJEXT2-samp}}(\mathcal{B}, \mathcal{R})$ by the same quantity. Since $\mathcal{X} = \text{F.Dom} = \{0, 1\}^n$, we have $1/|\mathcal{X}| = 2^{-n}$. Finally, when the condition on challenge queries applies to $\mathcal{B}$, the algorithm $\mathcal{A}$ never queries the challenge oracle, so neither does $\mathcal{R}$. This proves the theorem. $\square$