Representation-Aware Kleptographic Attacks on Post-Quantum Cryptography

Sayan Das[1], Suraj Mandal[2], Thales B. Paiva[3], Prasanna Ravi[1], Debapriya Basu Roy[2], Anupam Chattopadhyay[1], and Shivam Bhasin[1]

[1] Nanyang Technological University, Singapore
[2] Indian Institute of Technology Kanpur, India
[3] University of São Paulo, Brazil
sayan005@e.ntu.edu.sg, surajm22@iitk.ac.in, thalespaiva@larc.usp.br, prasanna.ravi@ntu.edu.sg, dbroy@cse.iitk.ac.in, anupam@ntu.edu.sg, sbhasin@ntu.edu.sg

Abstract. Kleptographic subversions of cryptographic implementations, through Algorithm Substitution Attacks (ASAs), threaten the trust assumptions of black-box cryptosystems. This work studies such subversions in Post-Quantum Cryptography (PQC) through the lens of public-key representation. We formalize key-generation subversion with explicit public-key representations, distinguishing seed-compressed interfaces from expanded-key interfaces. This distinction captures a security-relevant role of seed compression beyond bandwidth and storage efficiency: public components regenerated from seeds can be checked against the specification, creating a regeneration barrier against certain expanded-component substitutions.

We further present the first kleptographic key-generation backdoor targeting HQC. Our construction instantiates the model in the expanded-key setting by building a BIKE-form in-band carrier that exploits structural similarities between HQC public-key formation and BIKE decoding instances. A subverter holding the BIKE decoding trapdoor can recover the randomness used to derive the victim's HQC secret state. We implement a dimension-aligned proof of concept and evaluate functional correctness, runtime overhead, and coarse black-box statistical tests. We further provide a conditional indistinguishability proof for subverted keys.

Moreover, we contrast this behaviour with ML-DSA, where recomputability from public seeds and secret-key material constrains analogous perturbation-style subversions. We also analyze a generic seed-level SETUP pattern as a remaining attack surface, showing that seed compression blocks certain expanded-component carriers but does not by itself eliminate seed-generation subversion. Overall, our results support the view that public-key representation should be treated as part of the kleptographic assurance boundary for PQC implementations.

Keywords: Kleptography · Backdoors · Algorithm Substitution · Post-Quantum Cryptography · HQC · ML-DSA

1 Introduction

Kleptography concerns the use of *cryptography against cryptography* [27]: an adversary embeds a subliminal mechanism into an otherwise correct-looking cryptographic implementation. In a typical kleptographic setting, secret information is exfiltrated through publicly observable artifacts such as ciphertexts, signatures, public keys, or protocol transcripts without changing the apparent functionality of the primitive. Such attacks naturally arise in a *black-box* threat model, where the user observes only an algorithm's external inputs and outputs. These subverted systems are referred as Secretly Embedded Trapdoors with Universal Protections (SETUPs) [26], which directly exploit the trust assumption of black-box cryptosystems.

Despite the conceptual clarity of this threat, kleptography and related algorithmic backdoors received limited sustained attention in mainstream applied-cryptography practice until incident-driven concerns renewed interest in systematic weakening of cryptographic systems and standards [3, 11, 19]. Subsequent episodes reinforced that backdoors can be introduced through subtle manipulations of cryptographic components, including random-number generation, parameter generation, and protocol logic [7, 15, 16].

Much of this literature and incident-driven analysis has focused on *classical* public-key settings: discrete-logarithm systems, elliptic-curve ecosystems, and legacy protocol stacks whose security critically depends on high-quality randomness and parameter generation [15, 16, 27]. The post-quantum transition changes this landscape. Standardized and near-standardized PQC schemes introduce implementation interfaces such as seed expansion, structured sampling, rejection logic, compression, and serialization.

Comparatively few works have examined how kleptographic subversion changes under the public-key representation choices used in modern PQC implementations. Recent works demonstrate practical ASA backdoors for lattice-based KEMs, including ML-KEM/Kyber, under realistic constraints [12, 18, 23]. These works show that seed-derived key-generation pipelines can be subverted, but they do not isolate public-key representation itself as a security boundary. In many PQC specifications, seed-compressed public keys are primarily motivated by bandwidth and storage efficiency; however, we observe that this representation also has a security consequence. If public components are regenerated from a seed, then a verifier can check whether the expanded public key is consistent with the prescribed expansion procedure. This regeneration check can rule out certain expanded-key substitutions that may otherwise be difficult to distinguish from a standalone public key.

This distinction becomes operationally relevant when public keys are serialized, certified, and archived in deployment infrastructures. For example, X.509 binds public keys to identities, and applications such as S/MIME rely on long-term certified encryption keys that may be reused and archived across deployment lifetimes [4, 20, 22]. Thus, public-key representation is not merely a syntactic choice: it affects the available checks against kleptographic subversion. This motivates the central question of this paper: ***What subversion opportunities***

arise when seed-compressed public-key interfaces are replaced by, or coexist with, expanded public-key representations?

We study this question using HQC [14] as a concrete code-based case study. Our main construction uses BIKE [2] as an attacker-only extraction mechanism and exploits the shared cyclic sparse-linear structure of HQC public-key formation and BIKE encryption. We then contrast this with ML-DSA [17], where public-key recomputability constrains analogous perturbation-style subversions. Finally, we analyze seed-level SETUPs to clarify that seed compression is not a complete defense: it blocks a class of expanded-component carriers, but a subverter may still attempt to backdoor the seed-generation process itself. Our contributions are as follows:

- **Formal Model for Representation-Aware ASAs.** We formalize key-generation subversion in the ASA/SETUP setting with explicit attention to public-key representation. In particular, we distinguish *seed-compressed* interfaces from *expanded-key* interfaces and show how this distinction affects both stealth and extraction. This model captures the role of seed compression not only as a bandwidth optimization, but also as a regeneration-based conformance check against certain expanded-component backdoors.
- **First Kleptographic Backdoor for HQC.** To the best of our knowledge, we present the first kleptographic key-generation backdoor targeting HQC. The construction applies in the expanded-key setting and uses BIKE as an attacker-only extraction mechanism. It replaces the honestly expanded HQC public component with a BIKE-form carrier, $h'_{\mathrm{HQC}} = r_1 + r_2 h_{\mathrm{BIKE}}$, derives the HQC secret state from (r_1, r_2), and enables recovery only for an adversary holding the corresponding BIKE decoding trapdoor. We implement a dimension-aligned proof of concept and evaluate functional correctness, runtime overhead, and coarse black-box statistical tests.
- **Conditional Stealth Reduction for Backdoor Construction.** Single-key public-output stealth of the subverted parity-balanced HQC key generation construction is reduced to an explicit carrier-pseudorandomness assumption in the classical random-oracle model, with the dependence between the public carrier and the derived secret state accounted for.
- **Recomputability-based kleptographic analysis of ML-DSA.** We analyze why perturbation-style key-generation backdoors that hide helper data in error-like terms do not directly transfer to ML-DSA. The reason is that ML-DSA exposes enough public seeds and secret-key material to support recomputation checks on the public key. This contrast illustrates that the feasibility of key-generation subversion depends on the scheme's public-key representation and recomputability interface, not only on the underlying hardness assumption. We further describe a generic seed-level SETUP pattern in which the public seed itself is made to carry a trapdoor relation to the secret-generation seed.

Table 1. Related work on kleptographic attacks on cryptographic primitives.

Area / Works	Target	Approach and Scope
Classical [1, 5, 6, 21, 27–29]	RSA, Rabin, DH	Exploit randomness or algebraic structure to embed recoverable secrets, illustrating trade-offs between undetectability and leakage bandwidth.
Lattice backdoors [10, 24, 25]	NTRU, RLWE	Embed leakage in ciphertexts or public-key components; applicability to KEM key-generation subversion depends on the construction.
Seed-based SETUP [8]	Seeded PQC	Correlate public and secret seeds to enable recovery; applicability depends on seed-generation and key-representation interfaces.
Kyber backdoors [12, 18, 23]	Kyber	Leak key-generation secrets through public keys using classical or post-quantum carriers, with undetectability analyses and hardware-oriented variants.
Classic McEliece [9]	Code-based KEM	Embed an encryption of the secret seed through permutation/support structure; the mechanism does not directly transfer to BIKE or HQC.
This work	HQC	Investigates a carrier in expanded public keys and the regeneration constraint imposed by seed-compressed interfaces.

2 Related Works

Research on cryptographic backdoors spans *kleptographic* subversions that preserve the external input-output behaviour of a primitive while covertly leaking secrets. A renewed urgency around the topic followed the Snowden disclosures, which highlighted that backdoors may arise not only through implementation-level compromise but also through the manipulation of cryptographic standards and parameter choices. Since then, a body of work has studied backdoor insertion in both classical and post-quantum cryptographic primitives. We summarise representative works in Table 1.

3 Preliminaries and Threat Model

We model a KEM by a tuple of algorithms $\Pi = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$, where

$$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda), \qquad (\mathtt{ct}, K) \leftarrow \mathsf{Encaps}(\mathtt{pk}), \qquad K \leftarrow \mathsf{Decaps}(\mathtt{sk}, \mathtt{ct}).$$

Correctness means that a key pair generated by KeyGen decapsulates honestly generated ciphertexts to the same shared key, except with the negligible or explicitly bounded failure probability allowed by the scheme. Our main construction targets the key-generation interface of the HQC KEM. We refer to signature schemes only as a contrast case, where the relevant distinction is whether public keys can be recomputed from public seeds and secret-key material.

3.1 Public-Key Representations

A recurring feature of modern PQC implementations is that the object used internally by the cryptographic algorithms need not be identical to the object

serialized as a public key. We therefore distinguish the *mathematical public key* $\mathtt{pk}$ from its *public-key representation* $\rho_{\mathtt{pk}}$. Let $\mathsf{ExpandPK}_{\Pi} : \mathcal{R}_{\Pi} \to \mathcal{PK}_{\Pi}$ be the deterministic expansion map from a serialized representation $\rho_{\mathtt{pk}} \in \mathcal{R}_{\Pi}$ to the public-key object $\mathtt{pk} \in \mathcal{PK}_{\Pi}$ used by the primitive. A representation is well formed if $\mathsf{WF}_{\Pi}(\rho_{\mathtt{pk}}) = 1$, where WF_{Π} captures the public syntactic and domain checks required by the specification or deployment profile.

This distinction is central for seeded PQC designs. A public key may be represented compactly by a seed from which public components are regenerated, or it may be represented by the expanded components themselves. These two representations can support the same mathematical primitive but provide different degrees of freedom to a subverted implementation.

Seed-compressed interface. In a seed-compressed interface, the public representation contains a public seed seed_h and auxiliary public data v:

$$\rho_{\mathtt{pk}}^{\mathsf{seed}} = (\mathsf{seed}_h, v), \qquad \mathsf{ExpandPK}_{\Pi}(\rho_{\mathtt{pk}}^{\mathsf{seed}}) = (\mathsf{Expand}(\mathsf{seed}_h), v).$$

The public component $h = \mathsf{Expand}(\mathsf{seed}_h)$ can therefore be regenerated and checked against the specification. Thus, seed compression is not merely a bandwidth optimization: it also provides a conformance handle. A subverted implementation cannot freely choose an arbitrary expanded value $h^{\star}$ unless it can also provide a seed seed_h such that $\mathsf{Expand}(\mathsf{seed}_h) = h^{\star}$.

Expanded-key interface. In an expanded-key interface, the public representation directly contains the expanded component:

$$\rho_{\mathtt{pk}}^{\mathsf{exp}} = (h, v), \qquad \mathsf{ExpandPK}_{\Pi}(\rho_{\mathtt{pk}}^{\mathsf{exp}}) = (h, v).$$

Here, the regeneration check available in the seed-compressed interface is absent. A subverted implementation may therefore choose h directly, subject only to syntactic validity, functional correctness, and distributional plausibility. Our HQC construction targets precisely this expanded-key setting.

Observation and validation access. Our empirical black-box evaluation observes serialized public outputs only. It excludes secret keys, internal randomness, and execution traces. We separately discuss key-owner validation, where a trusted checker can access secret-key material. Results obtained under this stronger access model are not public-only detection results. The extraction public key may be known to the evaluator; the extraction secret remains with the subverter.

3.2 Representation-Relevant Subversion Surfaces

PQC key generation often consists of a pipeline that maps short randomness into structured public and secret objects. We use the term *subversion surface* for an internal step whose output is reflected, directly or indirectly, in public artifacts. The following surfaces are most relevant to the distinction between seed-compressed and expanded public-key interfaces.

– **Seed and expansion surfaces.** Seeds are expanded through XOFs or PRGs to generate public parameters, matrices, or ring elements. In a seed-compressed interface, these expansions can be recomputed and checked. In an expanded-key interface (such as MLKEM in X.509 [22]), the expanded object may be accepted directly, which increases the subverter's freedom. A separate seed-level SETUP may instead attempt to backdoor the seed-generation process itself.
– **Structured sampling surfaces.** Many PQC schemes sample sparse vectors, small coefficients, or bounded-noise objects from carefully specified distributions. A substituted sampler can correlate these values with a trapdoor while preserving functional correctness. Our HQC construction uses this surface by deriving secret material from recoverable sparse randomness.
– **Encoding and representation surfaces.** Public keys are serialized using scheme-specific encodings. The choice between seed-compressed and expanded public-key representations changes the available checks: canonical seed expansion constrains the public key, whereas expanded encodings rely mainly on well-formedness and distributional plausibility.

These surfaces are not presented as an exhaustive taxonomy. Rather, they identify the implementation interfaces needed for the HQC construction, the generic seed-level backdoor, and the contrast with recomputable schemes such as ML-DSA. The next section formalizes these distinctions through stealth and extraction notions over public-key representations.

4 Formalization of Key-Generation Subversion

We formalize key-generation subversion with explicit public-key representations. The representation determines which public components an implementation may output directly and which must be reconstructed through a prescribed expansion procedure. Let Π be a public-key primitive. Honest key generation with explicit randomness is written as

$$(\rho_{\mathtt{pk}}, \mathtt{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda; r), \qquad \mathtt{pk} = \mathsf{ExpandPK}_\Pi(\rho_{\mathtt{pk}}),$$

where $r \leftarrow\!\!\$\ \{0,1\}^{m(\lambda)}$. The predicate $\mathsf{WF}_\Pi(\rho_{\mathtt{pk}})$ checks public syntactic and domain requirements. Separately, $\mathsf{Corr}_\Pi(\mathtt{pk}, \mathtt{sk})$ denotes the binary outcome of a functional correctness experiment checking agreement between honest encapsulation and decapsulation for KEMs; and the verification of an honestly generated signature for Digital Signature Algorithms (DSAs).

4.1 Subverted Key Generation

An extraction setup algorithm samples

$$(\tau_{\mathsf{pub}}, \tau_{\mathsf{sec}}) \leftarrow \mathsf{Setup}(1^\lambda).$$

The substituted generator contains only the public component τ_{pub}, while the extraction secret τ_{sec} remains with the subverter:

$$(\widetilde{\rho}_{\mathtt{pk}}, \widetilde{\mathtt{sk}}) \leftarrow \mathsf{KeyGen}'(1^\lambda; r, \tau_{\mathsf{pub}}), \qquad \widetilde{\mathtt{pk}} = \mathsf{ExpandPK}_{\Pi}(\widetilde{\rho}_{\mathtt{pk}}).$$

We quantify functionality preservation by a failure bound δ_{func}:

$$\Pr\Big[\mathsf{WF}_{\Pi}(\widetilde{\rho}_{\mathtt{pk}}) = 1 \wedge \mathsf{Corr}_{\Pi}(\widetilde{\mathtt{pk}}, \widetilde{\mathtt{sk}}) = 1\Big] \geq 1 - \delta_{\mathrm{func}}.$$

The probability includes setup, key-generation, and correctness-experiment randomness. The bound must be justified for the substituted construction; it is not inherited automatically from the honest implementation.

Format Preservation. A substitution is *format-preserving* relative to $\mathcal{R}_{\Pi}$ when its public outputs satisfy the same length, encoding, and domain requirements as honest outputs in that representation. A substitution that changes these requirements is *format-changing*. It is to be noted that our HQC backdoor construction concerns expanded-key interfaces, rather than replacement of a seeded public-key encoding by an expanded byte string.

4.2 Regeneration Barrier for Seed-Compressed Keys

Suppose a substitution requires a public component to equal a chosen carrier $h^\star$. An expanded-key interface permits the output of $h^\star$ directly, subject to the interface's validation rules. A seed-compressed interface instead requires a seed seed_h satisfying

$$\mathsf{Expand}(\mathsf{seed}_h) = h^\star.$$

We call this constraint the *regeneration barrier*. A carrier chosen independently in the expanded domain cannot be represented through the seeded interface without supplying a corresponding seed. The difficulty of finding such a seed depends on the expansion function and the carrier-generation strategy; we do not claim a general impossibility result. The constraint also does not exclude seed-level subversions that operate through the prescribed expansion procedure.

4.3 Stealth and Extraction

We distinguish public-output stealth from privileged recovery of targeted secret material. Our formal stealth result concerns one challenge public key. The evaluator observes its public representation and the extraction public key, but not secret keys, key-generation randomness, internal state, or timing. Where a random-oracle model is invoked, the evaluator also has classical access to the specified oracle.

Definition 1 (Single-Key Public-Output Stealth). *Given a public-key representation $\mathcal{R}_{\Pi}$, the following experiments are considered parameterized by a bit $b \in \{0,1\}$. First, an extraction-key pair is sampled,*

$$(\tau_{\mathsf{pub}}, \tau_{\mathsf{sec}}) \leftarrow \mathsf{Setup}(1^\lambda),$$

and a challenge key pair is generated according to b,

$$(\rho_{\boldsymbol{pk}}^{(b)}, \boldsymbol{sk}^{(b)}) \leftarrow \begin{cases} \mathsf{KeyGen}^{H}(1^{\lambda}), & b = 0, \\ \mathsf{KeyGen}'^{H}(1^{\lambda}; \tau_{\mathsf{pub}}), & b = 1. \end{cases}$$

A probabilistic polynomial-time distinguisher $\mathcal{D}$ is then given $(1^{\lambda}, \tau_{\mathsf{pub}}, \rho_{\boldsymbol{pk}}^{(b)})$ together with classical oracle access to H, and outputs a bit. The stealth advantage *of $\mathcal{D}$ is*

$$\begin{aligned} \mathsf{Adv}_{\mathcal{D}}^{\mathrm{st}}(\lambda) = \Big| & \Pr\left[\mathcal{D}^{H}(1^{\lambda}, \tau_{\mathsf{pub}}, \rho_{\boldsymbol{pk}}^{(1)}) = 1\right] \\ & - \Pr\left[\mathcal{D}^{H}(1^{\lambda}, \tau_{\mathsf{pub}}, \rho_{\boldsymbol{pk}}^{(0)}) = 1\right]\Big|, \end{aligned}$$

where the probability is taken over Setup*, the random oracle H, key generation, and the randomness of $\mathcal{D}$.*

The substituted generator is single-key publicly stealthy for $\mathcal{R}_{\Pi}$ *if* $\mathsf{Adv}_{\mathcal{D}}^{\mathrm{st}}(\lambda)$ *is negligible for every classical probabilistic polynomial-time distinguisher $\mathcal{D}$.*

Disclosure of τ_{pub} ensures that secrecy of the embedded extraction public key is not required. Security for multiple challenge keys under a shared setup, or for distinguishers given secret keys or internal state, is not included in Definition 1.

Definition 2 (Extraction). *Let* Target *be a deterministic function of the generated secret state. An extractor $\mathcal{A}$ is a probabilistic polynomial-time algorithm whose output is obtained as*

$$\widehat{z} \leftarrow \mathcal{A}^{H}(1^{\lambda}, \tau_{\mathsf{pub}}, \tau_{\mathsf{sec}}, \widetilde{\rho}_{\boldsymbol{pk}}),$$

where

$$(\widetilde{\rho}_{\boldsymbol{pk}}, \widetilde{\boldsymbol{sk}}) \leftarrow \mathsf{KeyGen}'^{H}(1^{\lambda}; \tau_{\mathsf{pub}}).$$

The extraction success probability is defined by

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{ex}}(\lambda) = \Pr\left[\widehat{z} = \mathsf{Target}(\widetilde{\boldsymbol{sk}})\right],$$

with probability taken over setup, the random oracle, key generation, and extraction.

For full secret-key recovery, $\mathsf{Target}(\widetilde{\mathbf{sk}}) = \widetilde{\mathbf{sk}}$. In our ideal HQC construction, the effective secret state is (x, y) and this pair is the extraction target. The carrier errors (r_1, r_2) are intermediate values whose recovery allows reconstruction of that target. This does not assert recovery of every field in a serialized KEM secret key.

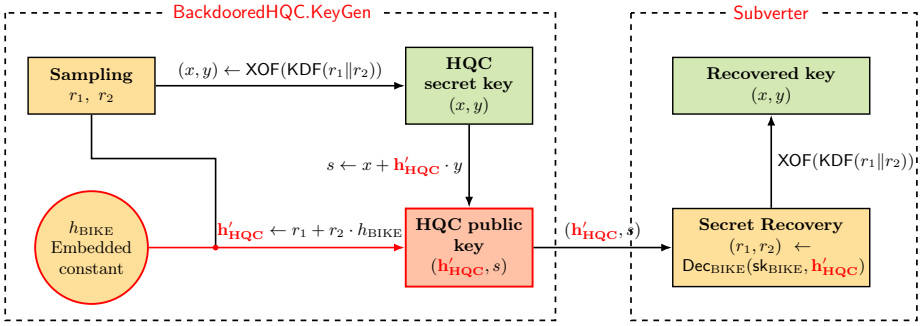

Figure 1. Overview of the expanded-key HQC backdoor.

5 Public-key Subversion of HQC using BIKE

We now instantiate the formal model for HQC in the expanded-key setting. The construction exploits the fact that, once the expanded public component is accepted directly, a subverted implementation can choose this component rather than deriving it from a public seed. The seed-compressed HQC interface blocks this choice through the regeneration barrier of Section 4.2; the expanded-key interface removes that check.

At a high level (Fig. 1), the substituted key generator replaces the honestly expanded HQC public component with an in-band carrier that has the algebraic form of a BIKE decodable object. The implementation embeds only an attacker public component h_{BIKE}, while the corresponding BIKE secret key remains with the subverter:

$$\tau_{\mathsf{pub}} = h_{\mathrm{BIKE}}, \qquad \tau_{\mathsf{sec}} = \mathtt{sk}_{\mathrm{BIKE}}.$$

The public output remains an HQC key of the expanded form (h, s), but h is chosen so that the subverter can later decode hidden sparse randomness from it.

5.1 Subverted Key Generation

The malicious key generator replaces the honest public-key expansion step with the BIKE-induced construction above. It then derives the victim's HQC secret state from the embedded vectors and completes the remainder of HQC key generation as usual. Algorithm 1 summarizes the substituted routine. The resulting public key remains syntactically valid in any interface that accepts the expanded form (h, s). At the same time, the attacker can interpret the published h'_{HQC} as a BIKE-form decoding instance under h_{BIKE}.

Once the maliciously generated public key is published, the adversary extracts the hidden state by decoding h'_{HQC} with the BIKE secret key (Algorithm 2). This yields the embedded vectors (r_1, r_2), from which the victim's HQC secret state can be reconstructed deterministically. As extraction only uses the public output and the attacker-held BIKE trapdoor, it requires no further interaction with the victim beyond observing the maliciously generated public key.

Require: Embedded public trapdoor $\tau_{\mathsf{pub}} = h_{\mathrm{BIKE}}$

$\omega \xleftarrow{\$} \{0,1\}^{256}$
$(r_1, r_2) \leftarrow \mathsf{SampleBIKEError}(\omega)$
$h'_{\mathrm{HQC}} \leftarrow r_1 + r_2 h_{\mathrm{BIKE}}$
$\mathsf{seed}_{\mathsf{sk}} \leftarrow \mathsf{KDF}(r_1 \parallel r_2)$
$(x, y) \leftarrow \mathsf{SampleHQCSecret}(\mathsf{seed}_{\mathsf{sk}})$
$s \leftarrow x + h'_{\mathrm{HQC}} y$
$\rho^{\mathsf{exp}}_{\mathsf{pk}} \leftarrow (h'_{\mathrm{HQC}}, s)$
$\mathsf{sk} \leftarrow \mathsf{PackHQCSecret}(x, y, \mathsf{seed}_{\mathsf{sk}})$
return $(\rho^{\mathsf{exp}}_{\mathsf{pk}}, \mathsf{sk})$

Algorithm 1. Subverted HQC Key Generation in the Expanded-Key Interface

Require: Expanded HQC component h'_{HQC}, secret trapdoor $\tau_{\mathsf{sec}} = \mathtt{sk}_{\mathrm{BIKE}}$

$(e_0, e_1) \leftarrow \mathsf{Decode}_{\mathrm{BIKE}}(\mathtt{sk}_{\mathrm{BIKE}}, h'_{\mathrm{HQC}})$
$r_1 \leftarrow e_0, \quad r_2 \leftarrow e_1$
$\mathsf{seed}_{\mathsf{sk}} \leftarrow \mathsf{KDF}(r_1 \parallel r_2)$
$(x, y) \leftarrow \mathsf{SampleHQCSecret}(\mathsf{seed}_{\mathsf{sk}})$
return (x, y)

Algorithm 2. Subverter Extraction from the HQC Public Component

5.2 Implementation

To evaluate the practical feasibility of the construction, we built a proof-of-concept[4] by modifying the HQC and BIKE reference implementations. The goal of the prototype is not to model a standardized BIKE deployment, but to demonstrate that the algebraic carrier can be realized end-to-end, that honest users can continue to operate the compromised HQC implementation, and that simple black-box statistical tests do not immediately separate honest from backdoored outputs. It is to be noted that a hardware implementation is not necessary, as the embedding and extraction mechanisms are defined at the algorithmic level and can be instantiated and evaluated in software.

- **Parameter Alignment.** The extraction phase requires the published HQC component h'_{HQC} to be a BIKE-form decodable instance under the attacker's BIKE secret key. For this reason, we modified the BIKE reference implementation so that its block-length parameter matches the HQC-1 ring dimension. Concretely, we changed `R_BITS` in `defs.h` to 17669, thereby setting $N = 17669$ and aligning BIKE with HQC-1 at the ring level. This modification is part of the attacker-controlled prototype and is used only to obtain a dimension-compatible carrier for the proof of concept.
- **Backdoored Key Generation.** We inserted the substituted routine directly into the HQC key-generation path in `hqc.c`. At the point where the honest implementation would normally expand the public component from a seed using SHAKE-256, the modified routine instead imports the attacker-controlled BIKE public key h_{BIKE}, samples sparse vectors r_1 and r_2, and computes $h'_{\mathrm{HQC}} = r_1 + r_2 \cdot h_{\mathrm{BIKE}}$ with varying parities to match honest h. The implementation then derives the victim's HQC secret vectors (x, y) from

[4] https://anonymous.4open.science/r/klepto-hqc-space-6566

Table 2. Empirical evaluation of 10^6 honest and 10^6 substituted expanded HQC-1 public keys. KL values are in bits and shown at the reported precision. Chi-squared p-values are unadjusted; all four Holm-adjusted values equal 1.

Measure	Observed result	Scope / interpretation
Pooled-byte KL	$D_{\mathrm{KL}}(P_H \| U) \approx 10^{-6}$ $D_{\mathrm{KL}}(P_B \| U) \approx 10^{-6}$ $D_{\mathrm{KL}}(P_H \| P_B) = 0.000000$ (rounded)	Close agreement of pooled byte-frequency histograms; rounding to zero does not establish equality.
Pooled-byte K–S	$D_{\mathrm{KS}} \approx 1.3 \times 10^{-5}$	Small empirical cumulative-distribution difference; reported descriptively without a calibrated significance claim.
Chi-squared: h	Honest: $\chi^2 = 243.967$, $p = 0.6794$ Substituted: $\chi^2 = 255.586$, $p = 0.4779$	Uniformity of the randomly selected full-byte marginal is not rejected for either generator after Holm correction.
Chi-squared: s	Honest: $\chi^2 = 215.944$, $p = 0.9638$ Substituted: $\chi^2 = 227.037$, $p = 0.8956$	Uniformity of the randomly selected full-byte marginal is not rejected for either generator after Holm correction.

$(r_1 \parallel r_2)$ and computes the corresponding syndrome component s in the usual way.

– **Operational Correctness.** The substituted public component h'_{HQC} and the corresponding secret state are generated from the same backdoor randomness (r_1, r_2), preserving the key-consistency relation required for decapsulation. Although honest public-seed expansion is bypassed, ciphertexts generated under the substituted public key can be correctly decapsulated by the victim implementation. The correctness claim is therefore restricted to black-box functionality under the expanded-key interface.

5.3 Statistical Evaluation

To assess empirical black-box detectability, we analyze 10^6 honest and 10^6 substituted expanded HQC-1 public keys. Each sample contains the serialized pair (h, s), occupying 4418 bytes, yielding 4.418 GB per class. These sizes refer to the expanded representation evaluated here, rather than HQC's specified seeded representation. We examine pooled byte-frequency divergence, empirical cumulative-distribution differences, and sampled byte-marginal uniformity. Table 2 summarizes the results.

These measurements characterize the selected byte-level features. They do not establish statistical or computational indistinguishability of complete public keys, and they do not exclude distinguishers exploiting whole-key properties or dependencies between h and s.

Execution Cost. We also measured runtime overhead over batches of 10,000 operations. The additional work is concentrated almost entirely in key generation, where the prototype must compute the extra product $r_2 \cdot h_{\mathrm{BIKE}}$. In our

measurements, HQC key generation increased from approximately 578 μs to 1193 μs, corresponding to an overhead of about 2.06×. By contrast, we observed no meaningful overhead in the online operations: encapsulation remained around 1158 μs and decapsulation around 616 μs. The overall cycle runtime increased by 1.26×. Hence, for a black-box system, where runtimes of different modules cannot be distinguished, the overhead on the runtime is acceptable. This behaviour is consistent with the attack model. The expensive step is paid only when a malicious key is generated, whereas the victim's subsequent use of that key remains close to the honest implementation from a coarse runtime perspective.

5.4 Conditional Stealth and Extraction

Single-key public-output stealth, as specified in Definition 1, is established below for an idealized expanded-key construction under an explicit carrier-pseudorandomness assumption. All parameters depend on the security parameter λ.

Construction and Idealization. Each element $v \in R = \mathbb{F}_2[X]/(X^n - 1)$ is represented uniquely as

$$v(X) = \sum_{i=0}^{n-1} v_i X^i, \qquad v_i \in \mathbb{F}_2.$$

Its Hamming weight is defined by $\mathrm{wt}(v) = |\{i : v_i = 1\}|$. The parity map $\pi : R \to \mathbb{F}_2$ is defined by

$$\pi(v) = v(1) = \sum_{i=0}^{n-1} v_i = \mathrm{wt}(v) \bmod 2,$$

where the sum is evaluated in $\mathbb{F}_2$. Thus, $\pi(v) = 0$ for even-weight vectors and $\pi(v) = 1$ for odd-weight vectors. Evaluation at $X = 1$ is well defined on R, since $(X^n - 1)|_{X=1} = 0$. For $p \in \{0, 1\}$, the parity class

$$R_p = \{v \in R : \pi(v) = p\}$$

is the set of vectors of parity p. Each class contains 2^{n-1} elements. For an integer $j \in \{0, \ldots, n\}$, the fixed-weight set is defined by

$$S_j = \{v \in R : \mathrm{wt}(v) = j\}.$$

Let a be the prescribed error-weight parameter and $\beta \in \{0, 1\}$ the uniformly sampled branch bit. The admissible error-pair space for branch β is defined by

$$\mathcal{W}_\beta = S_{a-1+\beta} \times S_a.$$

An ordered pair $W = (r_1, r_2)$ is sampled uniformly from $\mathcal{W}_\beta$; equivalently, r_1 and r_2 are sampled independently and uniformly from $S_{a-1+\beta}$ and S_a, respectively.

For the public extraction key h_B, the carrier map is defined by

$$F_{h_B} : \mathcal{W}_0 \cup \mathcal{W}_1 \to R, \qquad F_{h_B}(r_1, r_2) = r_1 + r_2 h_B,$$

where addition and multiplication are performed in R. The notation $F_{h_B}(W)$ is therefore shorthand for $F_{h_B}(r_1, r_2)$. Since $\pi(h_B) = 1$ and evaluation at $X = 1$ preserves addition and multiplication,

$$\pi(F_{h_B}(W)) = \pi(r_1) + \pi(r_2)\pi(h_B) = (a - 1 + \beta) + a \equiv 1 - \beta \pmod 2.$$

For $a = 75$, the two branches have weights $(74, 75)$ and $(75, 75)$, producing odd- and even-parity carriers, respectively.

Carrier assumption. For each branch $\beta \in \{0, 1\}$, let U_β denote a uniformly sampled element of

$$R_{1-\beta} = \{v \in R : \pi(v) = 1 - \beta\},$$

independently of the extraction-key setup. Thus, U_0 is sampled uniformly from the odd-parity polynomials, whereas U_1 is sampled uniformly from the even-parity polynomials. Let W be sampled uniformly from $\mathcal{W}_\beta$, independently of setup. Under the construction's actual setup distribution for h_B, the distributions

$$(h_B, \beta, F_{h_B}(W)) \quad \text{and} \quad (h_B, \beta, U_\beta)$$

are assumed to be computationally indistinguishable. In other words, a carrier generated from the prescribed error pair is assumed to be computationally indistinguishable from a uniformly sampled polynomial of the same parity, even when h_B and β are disclosed.

The maximum distinguishing advantage over algorithms with running time at most T is denoted by $\epsilon_{\mathsf{car},\beta}(T)$. This assumption is specific to the chosen ring, extraction-key setup, and error distribution; it is not implied by standard BIKE security without a separate reduction.

Proposition 1 (Conditional Public-Output Stealth). *Let $\mathcal{D}$ be a classical probabilistic polynomial-time distinguisher making at most q_H random-oracle queries. For the idealized construction,*

$$\mathsf{Adv}^{\mathrm{st}}_{\mathcal{D}}(\lambda) \le \epsilon_{\mathsf{car},0}(T) + \epsilon_{\mathsf{car},1}(T) + \frac{\eta_0 + \eta_1}{2}, \text{ where } \eta_\beta = \min\left\{1, \frac{|\mathcal{W}_\beta|}{2^{n-1}}\right\}.$$

The quantity η_β bounds the probability that a uniformly sampled polynomial of parity $1 - \beta$ admits an error-pair representation in $\mathcal{W}_\beta$. The running time T includes the execution of $\mathcal{D}$, random-oracle simulation, and evaluation of F_{h_B} on the relevant oracle queries. Consequently, Definition 1 is satisfied if both carrier advantages and η_0, η_1 are negligible.

Proof. Three games are considered, and the probability that $\mathcal{D}$ outputs 1 in Game i is denoted by p_i.

Game 0. The substituted experiment is executed without modification: β and W are sampled as specified above, $h^\star = F_{h_B}(W)$ is computed, and the secret pair is obtained as $(x^\star, y^\star) = H(W)$.

Game 1. The secret pair $(x^\star, y^\star)$ is instead sampled uniformly from $S_\omega \times S_\omega$, independently of H. The carrier is left unchanged, and $s^\star = x^\star + h^\star y^\star$ is computed.

The two games can be coupled so that their public keys and all oracle responses are identical until H is queried at W. Therefore,

$$|p_0 - p_1| \leq \Pr[\mathsf{Query}_W \text{ in Game 1}],$$

where Query_W denotes a query at the sampled carrier-error pair.

For a fixed branch β, and given a carrier challenge (h_B, β, c), Game 1 is simulated by independently sampling (x, y), supplying $(c, x + cy)$ to $\mathcal{D}$, and answering oracle queries by lazy sampling. Acceptance is declared if a query $W' \in \mathcal{W}_\beta$ satisfying $F_{h_B}(W') = c$ is observed.

For a uniform carrier challenge, acceptance requires $c \in F_{h_B}(\mathcal{W}_\beta)$. For every supported h_B, at most $|\mathcal{W}_\beta|$ distinct carriers can be generated from the admissible error pairs. Therefore,

$$\Pr[U_\beta \in F_{h_B}(\mathcal{W}_\beta) \mid h_B] = \frac{|F_{h_B}(\mathcal{W}_\beta)|}{2^{n-1}} \leq \eta_\beta.$$

The same bound is obtained after averaging over the setup.

Game 2. The carrier is replaced by $h^\star \leftarrow R_{1-\beta}$. The secret pair remains independent and uniform, and the syndrome is recomputed as $s^\star = x^\star + h^\star y^\star$. By a direct application of the carrier assumption,

$$|p_1 - p_2| \leq \frac{1}{2} \sum_{\beta=0}^{1} \epsilon_{\mathsf{car},\beta}(T).$$

Since β is uniform, the mixture of the two uniform parity classes is uniform over R. The carrier in Game 2 is therefore uniform and independent of the setup, secret pair, and random oracle. The ideal honest experiment of Definition 1 is consequently obtained.

The stated bound follows from

$$\mathsf{Adv}^{\mathrm{st}}_{\mathcal{D}}(\lambda) = |p_0 - p_2| \leq |p_0 - p_1| + |p_1 - p_2|.$$

Extraction. Let $\mathsf{Recover}_\tau$ denote recovery of the carrier errors using the extraction trapdoor, and we define

$$\delta_\beta = \Pr[\mathsf{Recover}_\tau(F_{h_B}(W)) \neq W \,|\, W \leftarrow \mathcal{W}_\beta],$$

where the probability is also taken over setup and any decoder randomness. Both declared failures and recovery of an incorrect error pair are included in δ_β.

The extractor of Definition 2 is implemented by recovering $\widehat{W}$ from the public carrier and returning $H(\widehat{W})$. Whenever $\widehat{W} = W$, the original secret pair is reconstructed exactly. Hence,

$$\mathsf{Adv}^{\mathrm{ex}}_{\mathcal{A}}(\lambda) \geq 1 - \frac{\delta_0 + \delta_1}{2}.$$

Scope and implementation correspondence. The proposition applies to a single public-key challenge and classical random-oracle queries. The entire secret-pair derivation is idealized by H, and uniform sampling from the prescribed fixed-weight sets is assumed. For a concrete seeded implementation, separate transitions to the ideal honest and substituted experiments are required. If their distinguishing bounds are denoted by $\epsilon_{\mathsf{seed},H}$ and $\epsilon_{\mathsf{seed},B}$, respectively, then

$$\mathsf{Adv}^{\mathrm{st}}_{\mathcal{D},\mathsf{concrete}} \leq \epsilon_{\mathsf{seed},H} + \mathsf{Adv}^{\mathrm{st}}_{\mathcal{D},\mathsf{ideal}} + \epsilon_{\mathsf{seed},B}.$$

These transition bounds must account for the implemented samplers, seed dependencies, and accessible derivation interfaces. Their negligibility is not established by the proposition. The statistical measurements reported in Subsection 5.3 are treated as empirical evaluations of selected public-output features. Neither the carrier assumption nor indistinguishability of the complete public-key distributions is established by those measurements.

6 Kleptographic Analysis of ML-DSA

The HQC construction in Section 5 shows that an expanded public-key component can become an in-band carrier when the implementation is allowed to choose that component directly. ML-DSA provides a useful contrast, invoking a recomputable public key map.

6.1 ML-DSA Key Generation and Public-Key Consistency

ML-DSA [17] is a module-lattice-based digital signature standard derived from CRYSTALS-Dilithium. It is defined over the quotient ring $R_q = \mathbb{Z}_q[X]/(X^n + 1)$, where q is the prime modulus and n is the polynomial degree. ML-DSA has three standardized parameter sets, commonly denoted ML-DSA-44, ML-DSA-65, and ML-DSA-87. Since our analysis concerns key-generation subversion, we focus only on the key-generation relation.

Algorithm 3 captures the dependency structure relevant to our analysis. The public matrix $\mathbf{A}$ is expanded from the public seed $seed_\rho$, while the short vectors $\mathbf{s}_1$ and $\mathbf{s}_2$ are expanded from $seed_{\rho'}$. The public vector is computed as $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$, and then decomposed by Power2Round into $(\mathbf{t}_1, \mathbf{t}_0)$. The public key contains $(seed_\rho, \mathbf{t}_1)$, while the secret key contains $seed_\rho$, $\mathbf{s}_1$, $\mathbf{s}_2$, and $\mathbf{t}_0$, among other values. This structure is the source of ML-DSA's contrast with the HQC expanded-key setting: the public component $\mathbf{t}_1$ is expanded in the certificate/public-key representation, but it is not freely choosable if the key pair is later checked for consistency. We further formalize the consistency property enforced by the ML-DSA key relation.

Definition 3 (Recomputable public-key map). *A scheme has a recomputable public-key map if there exists an efficient deterministic function* PubMap *such that for honestly generated keys,*

$$\boldsymbol{pk} = \mathsf{PubMap}(1^\lambda, \boldsymbol{sk}),$$

and $\mathsf{WF}_\Pi(\rho_{\boldsymbol{pk}}) = 1$ *for the corresponding representation.*

If a user or validation routine can perform the check

$$\mathtt{pk} \stackrel{?}{=} \mathsf{PubMap}(1^\lambda, \mathtt{sk}),$$

then any subversion that changes the public key without preserving this map is detected. In ML-DSA, this map is induced by the relation

$$\mathbf{A} = \mathsf{ExpandA}(\mathit{seed}_\rho), \qquad \mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2, \qquad (\mathbf{t}_1, \mathbf{t}_0) = \mathsf{Power2Round}_q(\mathbf{t}, d).$$

A validation routine with access to the secret key can therefore recompute $\mathbf{A}$, reconstruct $\mathbf{t}$, and check consistency with both the public component $\mathbf{t}_1$ and the secret component $\mathbf{t}_0$. This is the key distinction from the HQC expanded-key construction: an expanded ML-DSA public key is still constrained by a recomputable relation, whereas the HQC expanded component h_{HQC} can become an in-band carrier when it is accepted directly.

In the X.509 setting, this distinction remains relevant. RFC 9881 specifies ML-DSA public keys for PKI certificates and also describes private-key encodings, including seed-based private-key representation [13]. Thus, an ML-DSA certificate may contain an expanded public-key byte string, but key import, storage, or validation procedures may still support seed/expanded-key consistency checks. This differs from the HQC expanded-key setting studied earlier, where the chosen expanded component itself can become the carrier if no seed-regeneration check is enforced.

Limits of Direct Public-Key Perturbation in ML-DSA. Direct transfer of noise-based ML-KEM/Kyber backdoors [12, 18, 23] to ML-DSA is constrained by secret-key recomputability. Given the secret key, the relation $(\mathbf{t}_1, \mathbf{t}_0) = \mathsf{Power2Round}_q(\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2, d)$ can be checked against the public and stored secret components. Independent perturbations that violate this relation

procedure MLDSA.KeyGen
  $\mathit{seed} \leftarrow \{0,1\}^{256}$
  $(\mathit{seed}_\rho, \mathit{seed}_{\rho'}, K) \in \{0,1\}^{256} \times \{0,1\}^{512} \times \{0,1\}^{256} := H(\mathit{seed})$
  ▷ H is instantiated as SHAKE-256
  $\mathbf{A} \in R_q^{k \times \ell} := \mathrm{ExpandA}(\mathit{seed}_\rho)$ ▷ $\mathbf{A}$ is generated in the NTT domain as $\hat{\mathbf{A}}$
  $(\mathbf{s}_1, \mathbf{s}_2) \in S_\eta^\ell \times S_\eta^k := \mathrm{ExpandS}(\mathit{seed}_{\rho'})$
  $\mathbf{t} := \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$ ▷ Compute $\mathbf{A}\mathbf{s}_1$ as $\mathrm{NTT}^{-1}(\hat{\mathbf{A}} \cdot \mathrm{NTT}(\mathbf{s}_1))$
  $(\mathbf{t}_1, \mathbf{t}_0) := \mathrm{Power2Round}_q(\mathbf{t}, d)$
  $tr \in \{0,1\}^{256} := H(\mathit{seed}_\rho \,\|\, \mathbf{t}_1)$
  return $(pk = (\mathit{seed}_\rho, \mathbf{t}_1),\ sk = (\mathit{seed}_\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0))$

Algorithm 3. Key Generation Procedure of ML-DSA

procedure Backdoored.KeyGen
  Attacker installs it's public key pk_{bd}
  Random bitstream ρ is generated
  $Enc(\rho, pk_{bd}) = \sigma$
  $seed_A \leftarrow \rho$
  $seed_B \leftarrow G(\sigma)$
  $\hat{\mathbf{A}} \in R_q^{k \times k} \leftarrow \mathsf{Expand}(seed_A)$
  $\mathbf{s} \in R_q^k \leftarrow \mathsf{Sample}_B(seed_B, coin_s)$
  $\mathbf{e} \in R_q^k \leftarrow \mathsf{Sample}_B(seed_B, coin_e)$
  $\mathbf{t} = \mathbf{A} \circ \mathbf{s} + \mathbf{e}$
  Return $(pk = (seed_A, \mathbf{t}), sk = \mathbf{s})$

Algorithm 4. Generic Kleptographic Seed-Correlation Template

are therefore detectable, while modifications to $\mathbf{s}_2$ must also survive rounding to provide a public leakage channel. These checks require secret-key access and do not exclude consistently generated backdoored keys; the HQC construction likewise preserves its key-consistency relation. The contrast therefore concerns the constraints on specific embedding strategies, rather than general resistance to subversion. In particular, the seed controlling $\mathbf{s}_1$ and $\mathbf{s}_2$ may be maliciously generated while the public-key relation remains satisfied. This possibility motivates the following seed-level SETUP analysis: regeneration constrains independently chosen expanded components, but does not by itself exclude subversion of seed generation.

6.2 A Generic Seed-Level Backdoor and Its Detectability

The preceding sections separate two attack surfaces. In the HQC expanded-key setting, the attacker uses an expanded public component as an in-band carrier. In contrast, the ML-DSA analysis shows that recomputability can constrain direct perturbation of public components. A remaining generic possibility is therefore to attack the seed-generation process itself. This section analyzes such a seed-level SETUP pattern. A seed-level backdoor aims to correlate a public seed with a hidden secret-generation seed. Let $seed_A$ denote a public seed used to generate public material, and let $seed_B$ denote a seed used to sample secret or error material. A subverted implementation tries to make $seed_B$ recoverable by the subverter from public information, while preserving the apparent distribution of $seed_A$ and the derived public key.

Algorithm 4 gives a seed-correlation template. It should be interpreted carefully. For the construction to be a proper SETUP in the sense of Young and Yung, the implementation should contain only a public trapdoor component, while the secret extraction key remains with the subverter. Therefore, Enc should be understood as a trapdoor transform associated with a public extraction key, not as a symmetric encryption routine with a secret key hidden in the implementation. If a symmetric key is embedded in the implementation, then source-code or hardware inspection can reveal the key and allow anyone to exploit the leakage; such an instantiation is at best a weak black-box backdoor, not a strong SETUP.

There is also an extraction requirement. Since $seed_A$ is public, the subverter must be able to reconstruct the value used to derive $seed_B$ from the public output

and τ_{sec}. If Enc is randomized and its coins are not recoverable from the public seed, then the subverter cannot recompute σ from $seed_A$. If Enc is deterministic and publicly computable, then any observer may be able to compute the same σ, making the leakage non-exclusive. Thus, a strong seed-level SETUP requires a carefully designed trapdoor transform whose public output is indistinguishable from a valid public seed and whose hidden preimage is recoverable only to the subverter.

Detectability of Seed-Level Backdoors. Three detection regimes are distinguished:

- **Public-seed tests.** Biases, repetitions, or correlations may be detected through statistical tests on public seeds. Non-rejection does not establish indistinguishability.
- **Regeneration checks.** Inconsistencies between supplied expanded components and their seed-defined values can be detected. Maliciously generated seeds with consistent expansions are not excluded by these checks.
- **Implementation inspection.** Embedded extraction secrets or unauthorized derivation routines may be revealed through white-box inspection. For a strong SETUP, only public trapdoor material should be embedded; the extraction secret must be retained outside the victim implementation.

The seed-level SETUP template is conditional on compatible interfaces, indistinguishable public seeds, and exclusive trapdoor-assisted recovery; seed compression alone does not exclude such subversion.

7 Conclusion

This paper studied kleptographic subversion of PQC key generation through the lens of public-key representation. We instantiated a kleptographic attack on HQC by constructing a BIKE-form in-band backdoor in the expanded-key setting. The construction replaces the honestly expanded public component with $h'_{\mathrm{HQC}} = r_1 + r_2 h_{\mathrm{BIKE}}$, derives the HQC secret state from (r_1, r_2), and enables extraction for a subverter holding the BIKE decoding trapdoor. Our implementation demonstrates the feasibility of the carrier under dimension alignment and shows that coarse black-box statistical tests do not immediately distinguish the generated outputs at the evaluated scale. We further established a conditional stealth reduction to provide single-key public-output guarantees under an explicit carrier-pseudorandomness assumption in the classical random-oracle model. We also used ML-DSA as a contrast case, showing that recomputability from public seeds and secret-key material constrains analogous perturbation-style subversions, and analyzed seed-level SETUPs as a remaining generic attack surface.

Especially relevant for certificate-bound public keys in infrastructures such as X.509 where long-term public keys are serialized, distributed, and archived, the broader implication is that PQC assurance must account not only for the hardness assumptions of the standardized primitive, but also for the public key representation, serialization, and validation rules used in deployment.

References

1. Anderson, R., Vaudenay, S., Preneel, B., Nyberg, K.: The Newton channel. In: International Workshop on Information Hiding. pp. 151–156. Springer (1996), https://link.springer.com/chapter/10.1007/3-540-61996-8_38 Cited on page 4.
2. Aragon, N., Barreto, P., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Ghosh, S., Gueron, S., Güneysu, T., et al.: Bike: bit flipping key encapsulation (2022), https://bikesuite.org/files/v5.2/BIKE_Spec.2024.10.10.1.pdf Cited on page 3.
3. Ball, J., Borger, J., Greenwald, G.: Revealed: how US and UK spy agencies defeat internet privacy and security. https://www.theguardian.com/world/2013/sep/05/nsa-gchq-encryption-codes-security (2013), accessed 2026-01-27 Cited on page 2.
4. Boeyen, S., Santesson, S., Polk, T., Housley, R., Farrell, S., Cooper, D.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (May 2008). https://doi.org/10.17487/RFC5280, https://www.rfc-editor.org/info/rfc5280 Cited on page 2.
5. Crépeau, C., Slakmon, A.: Simple backdoors for RSA key generation. In: Cryptographers' Track at the RSA Conference. pp. 403–416. Springer (2003), https://link.springer.com/chapter/10.1007/3-540-36563-X_28 Cited on page 4.
6. Dorey, K., Chang-Fong, N., Essex, A.: Indiscreet logs: Diffie-Hellman backdoors in TLS. In: NDSS (2017), https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/indiscreet-logs-persistent-diffie-hellman-backdoors-tls/ Cited on page 4.
7. Green, M.: On the juniper backdoor – a few thoughts on cryptographic engineering. https://blog.cryptographyengineering.com/2015/12/22/on-juniper-backdoor/ (2015), accessed 2026-01-27 Cited on page 2.
8. Hemmert, T.: How to backdoor LWE-like cryptosystems. Cryptology ePrint Archive (2022), https://eprint.iacr.org/2022/1381 Cited on page 4.
9. Hemmert, T., May, A., Mittmann, J., Schneider, C.R.T.: How to backdoor (classic) mceliece and how to guard against backdoors. In: International Conference on Post-Quantum Cryptography. pp. 24–44. Springer (2022) Cited on page 4.
10. Kwant, R., Lange, T., Thissen, K.: Lattice klepto: Turning post-quantum crypto against itself. In: International Conference on Selected Areas in Cryptography. pp. 336–354. Springer (2017), https://link.springer.com/chapter/10.1007/978-3-319-72565-9_17 Cited on page 4.
11. Larson, J., Perlroth, N., et al.: Revealed: The NSA's secret campaign to crack, undermine internet encryption. https://www.propublica.org/article/the-nsas-secret-campaign-to-crack-undermine-internet-encryption (2013), accessed 2026-01-27 Cited on page 2.
12. Mandal, S., Ravi, P., Dhilipkumar, M., Basu Roy, D., Chattopadhyay, A.: Hardware implementation of a stealthy and lightweight backdoor for crystals-kyber in hybrid cryptosystems. ACM Transactions on Embedded Computing Systems (2026) Cited on pages 2, 4, and 16.
13. Massimo, J., Kampanakis, P., Turner, S., Westerbaan, B.: Internet X.509 Public Key Infrastructure – Algorithm Identifiers for the Module-Lattice-Based Digital Signature Algorithm (ML-DSA). RFC 9881 (Oct 2025). https://doi.org/10.17487/RFC9881, https://www.rfc-editor.org/info/rfc9881 Cited on page 16.

14. Melchor, C.A., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Bos, J., Deneuville, J.C., Dion, A., Gaborit, P., Lacan, J., Persichetti, E., Robert, J.M., Véron, P., Zémor, G.: Hamming Quasi-Cyclic: HQC (2025), https://pqc-hqc.org/doc/hqc-specification_2025-02-19.pdf Cited on page 3.
15. Menn, J.: Exclusive: Secret contract tied NSA and security industry pioneer. https://www.reuters.com/article/world/exclusive-secret-contract-tied-nsa-and-security-industry-pioneer-idUSBRE9BJ1C5/ (2013), accessed 2026-01-27 Cited on page 2.
16. National Institute of Standards and Technology: NIST removes cryptography algorithm from random number generator recommendations. https://www.nist.gov/news-events/news/2014/04/nist-removes-cryptography-algorithm-random-number-generator-recommendations (2014), accessed 2026-01-27 Cited on page 2.
17. National Institute of Standards and Technology: Module–lattice–based digital signature standard. Federal Information Processing Standards Publication FIPS 204, U.S. Department of Commerce, Washington, D.C. (2024). https://doi.org/10.6028/NIST.FIPS.204, https://doi.org/10.6028/NIST.FIPS.204 Cited on pages 3 and 15.
18. Ravi, P., Bhasin, S., Chattopadhyay, A., Aikata, A., Sinha Roy, S.: Backdooring post-quantum cryptography: Kleptographic attacks on lattice-based KEMs. In: Proceedings of the Great Lakes Symposium on VLSI 2024. pp. 216–221 (2024), https://dl.acm.org/doi/abs/10.1145/3649476.3660373 Cited on pages 2, 4, and 16.
19. Richelson, J.T.: The Snowden affair: Web resource documents the latest firestorm over the national security agency. National Security Archive Electronic Briefing Book No. 436, National Security Archive, The George Washington University (Sep 2013), https://nsarchive2.gwu.edu/NSAEBB/NSAEBB436/ Accessed: 2026-01-08 Cited on page 2.
20. Schaad, J., Ramsdell, B.C., Turner, S.: Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification. RFC 8551 (Apr 2019). https://doi.org/10.17487/RFC8551, https://www.rfc-editor.org/info/rfc8551 Cited on page 2.
21. Schneider, C.R.T.: Backdoors in cryptography. Ruhr University, Bochum, Germany (2024), https://d-nb.info/135557921X/34 Cited on page 4.
22. Turner, S., Kampanakis, P., Massimo, J., Westerbaan, B.: Internet X.509 Public Key Infrastructure - Algorithm Identifiers for the Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM). RFC 9935 (Mar 2026). https://doi.org/10.17487/RFC9935, https://www.rfc-editor.org/info/rfc9935 Cited on pages 2 and 6.
23. Xia, W., Wang, G., Gu, D.: Post-quantum backdoor for Kyber-KEM. In: Selected Areas in Cryptography (SAC 2024) (2024), preproceedings PDF: https://sacworkshop.org/SAC24/preproceedings/XiaWangGu.pdf Cited on pages 2, 4, and 16.
24. Xiao, D., Yu, Y.: Klepto for ring-LWE encryption. The Computer Journal **61**(8), 1228–1239 (2018), https://academic.oup.com/comjnl/article-abstract/61/8/1228/5035449 Cited on page 4.
25. Yang, Z., Xie, T., Pan, Y.: Lattice klepto revisited. In: Proceedings of the 15th ACM Asia Conference on Computer and Communications Security. pp. 867–873 (2020), https://dl.acm.org/doi/abs/10.1145/3320269.3384768 Cited on page 4.

26. Young, A., Yung, M.: The dark side of "black-box" cryptography or: Should we trust capstone? In: Advances in Cryptology—CRYPTO'96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings 16. pp. 89–103. Springer (1996) Cited on page 2.
27. Young, A., Yung, M.: Kleptography: Using cryptography against cryptography. In: Advances in Cryptology—EUROCRYPT'97: International Conference on the Theory and Application of Cryptographic Techniques Konstanz, Germany, May 11–15, 1997 Proceedings 16. pp. 62–74. Springer (1997), https://link.springer.com/chapter/10.1007/3-540-69053-0_6 Cited on pages 2 and 4.
28. Young, A., Yung, M.: Bandwidth-optimal kleptographic attacks. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 235–250. Springer (2001), https://link.springer.com/chapter/10.1007/3-540-44709-1_20 Cited on page 4.
29. Young, A., Yung, M.: A space efficient backdoor in RSA and its applications. In: International Workshop on Selected Areas in Cryptography. pp. 128–143. Springer (2005), https://link.springer.com/chapter/10.1007/11693383_9 Cited on page 4.