On the Security of NTS-KEM in the Quantum Random Oracle Model

Varun Maram

Department of Computer Science,
ETH Zurich, Switzerland
vmaram@inf.ethz.ch

Abstract. NTS-KEM is one of the 17 post-quantum public-key encryption (PKE) and key establishment schemes remaining in contention for standardization by NIST. It is a code-based cryptosystem that starts with a combination of the (weakly secure) McEliece and Niederreiter PKE schemes and applies a variant of the Fujisaki-Okamoto (Journal of Cryptology 2013) or Dent (IMACC 2003) transforms to build an IND-CCA secure key encapsulation mechanism (KEM) in the classical random oracle model (ROM). Such generic KEM transformations were also proven to be secure in the quantum ROM (QROM) by Hofheinz et. al. (TCC 2017), Jiang et. al. (Crypto 2018) and Saito et. al. (Eurocrypt 2018). However, the NTS-KEM specification has some peculiarities which means that these security proofs do not directly apply to it.
This paper identifies a subtle issue in the IND-CCA security proof of NTS-KEM in the classical ROM, as detailed in its initial NIST second round submission, and proposes some slight modifications to its specification which not only fixes this issue but also makes it IND-CCA secure in the QROM. We use the techniques of Jiang et. al. (Crypto 2018) and Saito et. al. (Eurocrypt 2018) to establish our IND-CCA security reduction for the modified version of NTS-KEM, achieving a loss in tightness of degree 2; a quadratic loss of this type is believed to be generally unavoidable for reductions in the QROM (Jiang at. al., ePrint 2019/494). Following our results, the NTS-KEM team has accepted our proposed changes by including them in an update to their second round submission to the NIST process.

Keywords: code-based, KEM, quantum random oracle model, IND-CCA security, NIST standardization

1 Introduction

NIST's post-quantum cryptography (PQC) standardization project reached its second phase when, on 30th January 2019, a shortlist of 26 second-round candidate algorithms was announced – out of which 17 are public-key encryption (PKE) and key establishment schemes, and the rest are digital signature schemes [NIS19]. In a public-key setting, a key encapsulation mechanism (KEM) is considered to be a versatile cryptographic primitive, as it can be used for efficient black-box constructions of secure PKE (via the KEM-DEM paradigm [CS03]), key

exchange and authenticated key exchange schemes [BCNP08,FOPS01]. Hence, a majority of these 17 second-round submissions are proposals for KEMs.

Indistinguishability against chosen-ciphertext attacks (IND-CCA) is widely accepted as the standard security notion for KEMs and PKE schemes, but it is usually more difficult to prove than weaker notions of security such as indistinguishability (IND-CPA) and one-wayness (OW-CPA) against chosen-plaintext attacks. Therefore, most of the NIST KEM submissions employ some generic transformations, as studied by Dent [Den03] and Hofheinz et. al. [HHK17], to construct an IND-CCA secure KEM from a weakly (OW-CPA or IND-CPA) secure PKE. To be specific, these generic constructions are usually variants of the Fujisaki-Okamoto transformation [FO13], e.g., $\text{FO}^{\perp}$, $\text{FO}^{\not\perp}$, $\text{FO}^{\perp}_{m}$ and $\text{FO}^{\not\perp}_{m}$ (as named in [HHK17]). Figure 1 contains a description of the $\text{FO}^{\not\perp}_{m}$ transformation that, given hash functions $G(.)$ and $H(.)$, turns an OW-CPA secure PKE $(\mathsf{KGen}_{\text{PKE}}, \mathsf{Enc}, \mathsf{Dec})$ to an IND-CCA secure KEM $(\mathsf{KGen}_{\text{KEM}}, \mathsf{Encap}, \mathsf{Decap})$ – see Subsection 2.3 for definitions of PKEs and KEMs. (Also, $\mathsf{Enc}(\mathsf{pk}, \mathbf{m}; G(\mathbf{m}))$ denotes that $G(\mathbf{m})$ is used as random coins in the encryption of message $\mathbf{m}$ sampled from the message space $\mathcal{M}$.)

$\mathsf{KGen}_{\text{KEM}}$	$\mathsf{Encap}(\mathsf{pk})$	$\mathsf{Decap}(\mathbf{c}, \mathsf{sk}')$
1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}_{\text{PKE}}$	1 : $\mathbf{m} \leftarrow\!\!\$\ \mathcal{M}$	1 : $\hat{\mathbf{m}} = \mathsf{Dec}(\mathsf{sk}, \mathbf{c})$
2 : $\mathbf{z} \leftarrow\!\!\$\ \mathcal{M}$	2 : $\mathbf{c} = \mathsf{Enc}(\mathsf{pk}, \mathbf{m}; G(\mathbf{m}))$	2 : **if** $\mathsf{Enc}(\mathsf{pk}, \hat{\mathbf{m}}; G(\hat{\mathbf{m}})) = \mathbf{c}$
3 : $\mathsf{sk}' = (\mathsf{sk}, \mathbf{z})$	3 : $\mathbf{K} = H(\mathbf{m})$	3 : **return** $H(\hat{\mathbf{m}})$
4 : **return** $(\mathsf{pk}, \mathsf{sk}')$	4 : **return** $(\mathbf{K}, \mathbf{c})$	4 : **else return** $H(\mathbf{z} \mid \mathbf{c})$

Fig. 1. IND-CCA secure KEM = $\text{FO}^{\not\perp}_{m}[\text{PKE}, G, H]$.

The other three variants (namely $\text{FO}^{\perp}$, $\text{FO}^{\not\perp}$ and $\text{FO}^{\perp}_{m}$) have slight differences: the subscript m (without m, resp.) means that, in the corresponding transformation, the encapsulated key $\mathbf{K}$ is equal to $H(\mathbf{m})$ ($\mathbf{K} = H(\mathbf{m} \mid \mathbf{c})$, resp.), and the superscript $\perp$ ($\not\perp$, resp.) means explicit[1] (implicit, resp.) rejection of invalid ciphertexts during decapsulation.

Typically, the security of such schemes is analyzed (heuristically) in the *random oracle model* (ROM), introduced in [BR93], where a hash function is idealized as a publicly accessible random oracle. But as pointed out by Boneh et. al. [BDF+11], in a post-quantum setting, an adversary could evaluate a hash function on an arbitrary superposition of inputs. This is not captured in the ROM as an adversary is only given a *classical* access to the random oracle. In order to fully assess the post-quantum security of cryptosystems, the *quantum random oracle model* (QROM) was advocated in [BDF+11]. Here, the adversary is allowed to make quantum queries to the random oracle. The above generic

[1] In explicit rejection, the symbol "$\perp$" is returned (instead of a pseudorandom key $H(\mathbf{z} \mid \mathbf{c})$ as is the case in implicit rejection) for the decapsulation of invalid ciphertexts.

KEM transformations ($\mathrm{FO}^{\perp}$, $\mathrm{FO}^{\not\perp}$, $\mathrm{FO}^{\perp}_m$, $\mathrm{FO}^{\not\perp}_m$) were initially only analyzed in the ROM by Hofheinz et. al. [HHK17] but then later were proven to be secure in the QROM by Jiang et. al. [JZC+18] and Saito et. al [SXY18], giving confidence in the NIST KEM candidates that rely on these transformations.

NTS-KEM is a KEM proposal that is shortlisted by NIST for PQC standardization. It is also one of a handful of second-round candidates that are code-based, as it is based on the well-known McEliece cryptosystem [McE78]. NTS-KEM employs a transformation similar to the Fujisaki-Okamoto [FO13] (or Dent [Den03]) transforms to achieve IND-CCA security of its KEM in the ROM. In particular, the transformation looks similar to $\mathrm{FO}^{\not\perp}_m$ since, as will be detailed in Section 3, NTS-KEM does an implicit rejection of invalid ciphertexts during decapsulation, and when computing the encapsulated keys, the ciphertext is not included in the input to the hash function. But at the same time, NTS-KEM contains significant variations from $\mathrm{FO}^{\not\perp}_m$ in its specification, meaning that straightforward application of known QROM security proofs for FO-transformations ([JZC+18,SXY18]) do not work. One of these major variations from $\mathrm{FO}^{\not\perp}_m$ is that, during the encapsulation of keys, the message $\mathbf{m}$ to be encrypted is not sampled uniformly from the message space, but is determined from the *randomness* $\mathbf{e}$ that is used in the McEliece-type encryption function.

1.1 Our Contributions

In this paper, we make two contributions:

- We identify a flaw in the IND-CCA security proof for NTS-KEM in the ROM, as described in its initial NIST second round submission. We also propose some changes to the specification of NTS-KEM which, in addition to fixing the flaw, preserve the tightness of the intended ROM proof.
- We present a proof of IND-CCA security for the modified version of NTS-KEM in the QROM. On a high level, our proof is structurally similar to [JZC+18]'s QROM security proof for a *modular* variant of $\mathrm{FO}^{\not\perp}_m$, namely $\mathrm{U}^{\not\perp}_m$. At the same time, our proof needs to account for significant differences between $\mathrm{U}^{\not\perp}_m$ and the new NTS-KEM specification.

To be specific, we recommend a re-encryption step in the NTS-KEM decapsulation routine (similar in spirit to the FO-type transformations) to account for invalid ciphertexts that may not be implicitly rejected. This change not only fixes NTS-KEM's tight IND-CCA security proof in the classical ROM but also leads to an IND-CCA security reduction in the QROM, only incurring a quadratic loss w.r.t. degree of tightness. This loss might be impossible to avoid [JZM19].

In order to formulate a security proof in the QROM for the modified NTS-KEM, we consider a modular FO-transformation $\mathrm{U}^{\not\perp}_m$, as studied in [HHK17], that turns a *deterministic* OW-CPA secure PKE into an IND-CCA secure KEM in the ROM. The only difference between $\mathrm{U}^{\not\perp}_m$ and $\mathrm{FO}^{\not\perp}_m$ is the encryption of messages during key encapsulation and decapsulation – because the encryption algorithm of the underlying PKE in $\mathrm{U}^{\not\perp}_m$ is deterministic, there is no need for

random coins $G(\mathbf{m})$. It was later shown in [JZC+18,SXY18] that $\mathrm{U}_m^{\not\perp}$ enhances security (from OW-CPA to IND-CCA) in the QROM as well. So to devise a proof based on the $\mathrm{U}_m^{\not\perp}$ framework, we view the random bits of the encryption function used in NTS-KEM, i.e., the error vectors $\mathbf{e}$, as *messages*, and formulate a deterministic encryption function $\mathrm{Encode}(\mathsf{pk}, \mathbf{e})$ (see *Encapsulation*, Section 3) that takes these error vectors as inputs. By doing this, we found it necessary to work with a non-standard security notion called *error one-wayness* or EOW security, as introduced in [ACP+19a], which is an analogue to OW-CPA security but for schemes that mainly process error vectors. Then the challenge is to account for notable differences between the modified NTS-KEM scheme and $\mathrm{U}_m^{\not\perp}$ in the proof, which includes the fact that to derive the encapsulated keys $\mathbf{K}$ in NTS-KEM, the message $\mathbf{m}$ itself is not hashed but a modified version of it is.

It is worth mentioning that the NTS-KEM team has adopted our proposed changes. On 3rd December 2019, an updated specification of NTS-KEM [ACP+19b] was posted on the website `https://nts-kem.io/`. Hence, in the rest of this paper, "NTS-KEM" will be used to refer to the updated version of the second round submission to NIST's PQC standardization process, unless stated otherwise.

2 Preliminaries

2.1 Notation

In this section, we outline notation borrowed from [ACP+19a] regarding NTS-KEM. We denote by $\mathbb{F}_2$ the field with two elements, and by $\mathbb{F}_{2^m}$ an extension field of $\mathbb{F}_2$ with 2^m elements. If $\mathbb{F}$ is a field, then $\mathbb{F}[x]$ is the ring of univariate polynomials with coefficients in $\mathbb{F}$. We denote by $\mathbb{F}_2^n$ the n-dimensional vector space with entries in $\mathbb{F}_2$, and by $\mathbb{F}_2^{k\times n}$ the kn-dimensional vector space of matrices with k rows and n columns with entries in $\mathbb{F}_2$. We denote vectors of $\mathbb{F}_2^n$ in bold lowercase, for example $\mathbf{e} = (e_0, e_1, \ldots, e_{n-1}) \in \mathbb{F}_2^n$; and matrices of $\mathbb{F}_2^{k\times n}$ in bold uppercase, for example $\mathbf{G} \in \mathbb{F}_2^{k\times n}$. The *Hamming weight* of a vector $\mathbf{e}$ is the number of non-zero components in the vector and is denoted by $\mathrm{hw}(\mathbf{e})$. Given a vector $\mathbf{e}$ of length n over a field $\mathbb{F}$, and positive integers $\ell < k < n$, we adopt the following notation to denote the partition of $\mathbf{e}$ into three sub-vectors: $\mathbf{e} = (\mathbf{e}_a \mid \mathbf{e}_b \mid \mathbf{e}_c)$, where $\mathbf{e}_a \in \mathbb{F}^{k-\ell}$, $\mathbf{e}_b \in \mathbb{F}^{\ell}$ and $\mathbf{e}_c \in \mathbb{F}^{n-k}$. More generally, if $\mathbf{v} \in \mathbb{F}^{n_1}$ and $\mathbf{w} \in \mathbb{F}^{n_2}$ are vectors over $\mathbb{F}$, we will denote by $(\mathbf{v} \mid \mathbf{w})$ the vector in $\mathbb{F}^{n_1+n_2}$ constructed as the concatenation of $\mathbf{v}$ and $\mathbf{w}$. A *permutation vector* $\mathbf{p} = (p_0, p_1, \ldots, p_{n-1})$ is a permutation of the n elements $\{0, 1, \ldots, n-1\}$. Then given the sequence $\mathbf{b} = (b_0, b_1, \ldots, b_{n-1})$, we denote the permuted sequence $\mathbf{b}' = \mathbf{b} \cdot \mathbf{P} = \pi_{\mathbf{p}}(\mathbf{b})$ such that $b'_i = b_{p_i}$, and the inverse permutation is given by $\mathbf{b} = \mathbf{b}' \cdot \mathbf{P}^{-1} = \pi_{\mathbf{p}}^{-1}(\mathbf{b}')$ such that $b_{p_i} = b'_i$. We denote the length of a vector $\mathbf{x}$ by $|\mathbf{x}|$.

The security parameter is denoted by λ. Given a set X, we denote by $x \leftarrow\!\!\$\, X$ the operation of sampling an element $x \in X$ uniformly at random, and we denote the sampling according to some arbitrary distribution D by $x \leftarrow D$. We denote probabilistic computation of an algorithm A on input x by $y \leftarrow\!\!\$\, A(x)$. $A^{H(.)}$ implies that the algorithm has access to the oracle $H(.)$.

2.2 Quantum Random Oracle Model

We introduce some lemmas in the QROM that will be used to derive the main results of this paper.

Lemma 1. (Simulating a QRO, [Zha12, Theorem 6.1]) *Let $H(.)$ be an oracle drawn from the set of $2q$-wise independent functions uniformly at random. Then the advantage any quantum algorithm making at most q quantum queries to $H(.)$ has in distinguishing $H(.)$ from a truly random oracle is identically 0.*

Lemma 2. ([SY17, Lemma C.1]) *Let $g_z : \{0,1\}^\ell \to \{0,1\}$ denotes a function defined as $g_z(z) = 1$ and $g_z(z') = 0$ for all $z \neq z'$, and $g_\perp : \{0,1\}^\ell \to \{0,1\}$ denotes a function that returns 0 for all inputs. Then for any unbounded adversary $\mathcal{C}$ that issues at most q quantum queries to its oracle, we have*

$$|\Pr[\mathcal{C}^{g_z(.)}(.) \to 1 \mid z \leftarrow_\$ \{0,1\}^\ell] - \Pr[\mathcal{C}^{g_\perp(.)}(.) \to 1]| \leq q \cdot 2^{-\frac{\ell+1}{2}}$$

Lemma 3. (Generalized OW2H[2] lemma, [JZC+18, Lemma 3 condensed][3]) *Let oracles $\mathcal{O}_1(.), \mathcal{O}_2(.)$, input parameter inp and x be sampled from a joint distribution D, where $x \in \{0,1\}^n$ (the domain of $\mathcal{O}_1(.)$). Consider a quantum oracle algorithm $\mathcal{U}^{\mathcal{O}_1,\mathcal{O}_2}$ which makes at most q_1 queries to $\mathcal{O}_1(.)$ and q_2 queries to $\mathcal{O}_2(.)$. Denote $\ddot{\mathcal{O}}_1(.)$ to be a reprogrammed oracle such that $\ddot{\mathcal{O}}_1(x) = y$, for a uniformly random y in $\{0,1\}^\ell$, and $\ddot{\mathcal{O}}_1(.) = \mathcal{O}_1(.)$ everywhere else. Let $\mathcal{V}^{\ddot{\mathcal{O}}_1,\mathcal{O}_2}$ be an oracle algorithm that on input $(inp, x, \mathcal{O}_1(x))$ does the following: picks $i \leftarrow_\$ \{1,\ldots,q_1\}$, runs $\mathcal{U}^{\ddot{\mathcal{O}}_1,\mathcal{O}_2}(inp, x, \mathcal{O}_1(x))$ until the i-th query to $\mathcal{O}_1(.)$, measures the query in the computational basis and outputs the measurement outcome (when $\mathcal{U}$ makes less than i queries, $\mathcal{V}$ outputs $\perp \notin \{0,1\}^n$). Define the events E_1, E_2 and probability $P_\mathcal{V}$ as follows,*

$$\Pr[E_1] = \Pr[b' = 1 : (\mathcal{O}_1, \mathcal{O}_2, inp, x) \leftarrow D, y \leftarrow_\$ \{0,1\}^\ell, b' \leftarrow \mathcal{U}^{\mathcal{O}_1,\mathcal{O}_2}(inp, x, \mathcal{O}_1(x))]$$

$$\Pr[E_2] = \Pr[b' = 1 : (\mathcal{O}_1, \mathcal{O}_2, inp, x) \leftarrow D, y \leftarrow_\$ \{0,1\}^\ell, b' \leftarrow \mathcal{U}^{\ddot{\mathcal{O}}_1,\mathcal{O}_2}(inp, x, \mathcal{O}_1(x))]$$

$$P_\mathcal{V} = \Pr[x' = x : (\mathcal{O}_1, \mathcal{O}_2, inp, x) \leftarrow D, y \leftarrow_\$ \{0,1\}^\ell, x' \leftarrow \mathcal{V}^{\ddot{\mathcal{O}}_1,\mathcal{O}_2}(inp, x, \mathcal{O}_1(x))]$$

Then $|\Pr[E_1] - \Pr[E_2]| \leq 2q_1\sqrt{P_\mathcal{V}}$.

[2] The one-way to hiding (OW2H) lemma, introduced in [Unr14], provides a generic reduction from a hiding-style property (indistinguishability) to a one-wayness-style property (unpredictability) in the QROM.

[3] We are referring to the latest version of [JZC+18] on the Cryptology ePrint Archive – Report 2017/1096, Version 20190703 – which differs from the conference version in that Lemma 3 no longer requires $\mathcal{O}_1(x)$ to be independent from $\mathcal{O}_2(.)$. Also we would be working with a condensed version of the lemma where we do not need $\mathcal{O}_1(x)$ to be uniformly distributed for any fixed $\mathcal{O}_1(x')$ $(x' \neq x)$, $\mathcal{O}_2(.)$, inp and x.

2.3 Cryptographic Primitives

Definition 1. *A Public Key Encryption scheme (PKE) consists of the following triple of polynomial-time algorithms* $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$.

- *The Key Generation algorithm* KGen *takes as input a security parameter* 1^λ *and outputs a public/private key-pair* $(\mathsf{pk}, \mathsf{sk})$.
- *The Encryption algorithm* Enc *takes as input a public key* pk *and a valid message* $\mathbf{m}$*, and outputs a ciphertext* $\mathbf{c}$.
- *The Decryption algorithm* Dec *takes as input a ciphertext* $\mathbf{c}$ *and a private key* sk*, and outputs a message* $\mathbf{m}$ *(or an error message indicating a decryption failure).*

For example, McEliece [McE78] proposed a PKE scheme which is based on linear error-correcting codes. Let $\mathcal{C} = [n, k, d]_2$ be such a code over $\mathbb{F}_2$ of length n and dimension k, with minimal distance d. The code $\mathcal{C}$ is capable of correcting at most $\tau = \left\lfloor \frac{d-1}{2} \right\rfloor$ errors, and can be described by a generator matrix $\mathbf{G} \in \mathbb{F}_2^{k \times n}$. Then a vector $\mathbf{w} \in \mathbb{F}_2^k$ can be encoded as a codeword in $\mathcal{C}$ as $\mathbf{c} = \mathbf{w} \cdot \mathbf{G} \in \mathbb{F}_2^n$. Now the McEliece scheme could be described as follows.

KGen: Generate a *special* type of $[n, k, d]_2$ linear error-correcting code $\mathcal{C}_\mathcal{G}$[4] with generator matrix $\mathbf{G}' \in \mathbb{F}_2^{k \times n}$ and that is capable of correcting up to τ errors; this special code is defined by a polynomial $G(z) \in \mathbb{F}_{2^m}[z]$ of degree τ. Let $\mathbf{S}$ be a non-singular matrix in $\mathbb{F}_2^{k \times k}$ and $\mathbf{P}$ be a permutation matrix in $\mathbb{F}_2^{n \times n}$, both generated at random. Then, define $\mathbf{G} = \mathbf{S} \cdot \mathbf{G}' \cdot \mathbf{P}$. The public key is given by $\mathsf{pk} = (\mathbf{G}, \tau)$ and the private key is $\mathsf{sk} = (G(z), \mathbf{S}^{-1}, \mathbf{P}^{-1})$.

Enc: To encrypt a message $\mathbf{m} \in \mathbb{F}_2^k$, sample $\mathbf{e} \in \mathbb{F}_2^n$ with Hamming weight τ and output the ciphertext $\mathbf{c} = \mathbf{m} \cdot \mathbf{G} + \mathbf{e} \in \mathbb{F}_2^n$.

Dec: To recover the message $\mathbf{m}$, compute $\mathbf{c}' = \mathbf{c} \cdot \mathbf{P}^{-1} = \mathbf{m} \cdot \mathbf{S} \cdot \mathbf{G}' + \mathbf{e} \cdot \mathbf{P}^{-1}$, and decode $\mathbf{c}'$ using a decoder for $\mathcal{C}_\mathcal{G}$ to recover the permuted $\mathbf{e}$, and hence $\mathbf{m}' = (\mathbf{m} \cdot \mathbf{S}) \in \mathbb{F}_2^k$. Finally, recover $\mathbf{m} = \mathbf{m}' \cdot \mathbf{S}^{-1}$ and output $\mathbf{m}$.

The McEliece PKE scheme achieves a certain notion of security known as *one-wayness* (OW), relying on hardness of the well-known problem of decoding random linear codes. Roughly speaking, the notion states that an adversary cannot recover the underlying message $\mathbf{m}$ from a given ciphertext $\mathbf{c}$. The OW security notion is formalized in Figure 2, where we write $\{0,1\}^{\mathsf{poly}(\lambda)}$ for the message space, indicating that it consists of bit-strings of some length that depends on some polynomial function of the security parameter.

An adversary $\mathcal{A}$ is said to be a (t, ε)-adversary against OW security of a PKE scheme if that adversary causes the $\mathrm{OW}^{\mathcal{A}}_{\mathsf{Enc}}$ game to output "1" with probability at least ε (where $0 < \varepsilon \leq 1$) and runs in time at most t. A PKE scheme is said to be (t, ε)-secure with respect to a given security notion, such as OW security, if no (t, ε)-adversary exists for that notion.

Definition 2. *A Key Encapsulation Mechanism (KEM) consists of the following triple of polynomial-time algorithms* $(\mathsf{KGen}, \mathsf{Encap}, \mathsf{Decap})$.

[4] Known as a binary *Goppa* code, as described in [ACP+19a].

$\mathrm{OW}^{\mathcal{A}}_{\mathsf{Enc}}$

1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$
2 : $\mathbf{m} \leftarrow_\$ \{0,1\}^{\mathsf{poly}(\lambda)}$
3 : $\mathbf{c} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{m})$
4 : $\mathbf{m}' \leftarrow \mathcal{A}(1^\lambda, \mathsf{pk}, \mathbf{c})$
5 : **return** $(\mathbf{m}' = \mathbf{m})$

Fig. 2. OW-security game for PKE.

- *The Key Generation algorithm* KGen *takes as input a security parameter* 1^λ *and outputs a public/private key-pair* $(\mathsf{pk}, \mathsf{sk})$.
- *The Encapsulation algorithm* Encap *takes as input a public key* pk *and outputs an encapsulated key and ciphertext* $(\mathbf{K}, \mathbf{c})$.
- *The Decapsulation algorithm* Decap *takes as input a ciphertext* $\mathbf{c}$ *and a private key* sk*, and outputs a key* $\mathbf{K}$ *encapsulated in* $\mathbf{c}$ *(or an error message "*$\perp$*" indicating a decapsulation failure).*

Compared to OW security, the desired notion for a KEM or a PKE scheme is IND-CCA security, i.e. *indistinguishability under chosen ciphertext attacks*. In the KEM version of this security notion, informally, an adversary should not be able to decide whether a given pair $(\mathbf{K}, \mathbf{c}^*)$ is such that $\mathbf{c}^*$ encapsulates $\mathbf{K}$ or if $\mathbf{K}$ is a random key independent of $\mathbf{c}^*$. In addition, the adversary is also given access to a decapsulation oracle that returns the output of $\mathsf{Decap}(\mathbf{c}', \mathsf{sk})$ for any $\mathbf{c}' \neq \mathbf{c}^*$ (and where we assume the adversary never queries $\mathbf{c}^*$ to this oracle, to prevent trivial wins). We denote this capability of accessing an oracle by the adversary as $\mathcal{A}^{\mathsf{Decap}(\cdot,\mathsf{sk})}(1^\lambda, \mathsf{pk}, \mathbf{K}_b, \mathbf{c}^*)$ in Figure 3.

$\text{IND-CCA}^{\mathcal{A}}_{\mathsf{KEM}}$

1 : $b \leftarrow_\$ \{0,1\}$
2 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$
3 : $(\mathbf{K}_0, \mathbf{c}^*) \leftarrow \mathsf{Encap}(\mathsf{pk})$
4 : $\mathbf{K}_1 \leftarrow_\$ \{0,1\}^{|\mathbf{K}_0|}$
5 : $b' \leftarrow \mathcal{A}^{\mathsf{Decap}(\cdot,\mathsf{sk})}(1^\lambda, \mathsf{pk}, \mathbf{K}_b, \mathbf{c}^*)$
6 : **return** $(b' = b)$

Fig. 3. IND-CCA-security game for KEM.

Formally, a (t, ε)-adversary against the IND-CCA security of a KEM causes the above game to return "1" with probability at least $1/2 + \varepsilon$ (where $0 < \varepsilon \leq 1/2$) and runs in time at most t. We say that a KEM is (t, ε)-secure in the IND-CCA sense if no (t, ε)-adversary exists; NTS-KEM is IND-CCA secure in

the classical random oracle model with a tight relationship to the OW-security of the McEliece PKE.

Finally, a KEM (respectively, PKE scheme) is said to be *perfectly correct* if for any public/private key pair $(\mathsf{pk}, \mathsf{sk})$ generated by KGen, we have $\Pr[\mathsf{Decap}(\mathbf{c}, \mathsf{sk}) = \mathbf{K} \mid (\mathbf{c}, \mathbf{K}) \leftarrow \mathsf{Encap}(\mathsf{pk})] = 1$ (respectively, $\Pr[\mathsf{Dec}(\mathbf{c}, \mathsf{sk}) = \mathbf{m} \mid \mathbf{c} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{m})] = 1$ for any valid message $\mathbf{m}$). For example, the McEliece scheme is a perfectly correct PKE and NTS-KEM is a perfectly correct KEM as shown in [ACP+19a].

3 NTS-KEM Specification

NTS-KEM is a key encapsulation mechanism that can be seen as a mixture of the McEliece and Niederreiter PKE schemes [McE78,Nie86] combined with a transform similar to the Fujisaki-Okamoto [FO13] or Dent [Den03] transforms to achieve (tight) IND-CCA security in the classical ROM. We provide a higher-level overview of the scheme's three main operations – namely *Key Generation*, *Encapsulation* and *Decapsulation* – that is relevant to the main results of this paper (refer to [ACP+19a,ACP+19b] for a more detailed description). Most importantly, the description below also includes our proposed changes to the decapsulation routine.

In the following, (n, τ, ℓ) are public parameters where $n = 2^m$ denotes the length of codewords, τ denotes the number of errors that can be corrected by the code (see McEliece PKE scheme in Subsection 2.3) and ℓ denotes the length of the random key to be encapsulated. Also k is a value which is chosen such that $k = n - \tau m$ with $\ell < k < n$. NTS-KEM uses a pseudorandom bit generator $H_\ell(.)$ to produce ℓ-bit binary strings; the current version uses the SHA3-256 hash function [NIS15] to implement $H_\ell(.)$.

Key Generation: Without going into details on how the keys are generated, it is sufficient to know that an NTS-KEM public key is given by $\mathsf{pk} = (\mathbf{Q}, \tau, \ell)$ where $\mathbf{Q} \in \mathbb{F}_2^{k \times (n-k)}$ is a matrix used in the encryption of messages during encapsulation, and private key is defined as $\mathsf{sk} = (\mathbf{a}^*, \mathbf{h}^*, \mathbf{p}, \mathbf{z}, \mathsf{pk})$ where $\mathbf{a}^*, \mathbf{h}^* \in \mathbb{F}_{2^m}^{n-k+\ell}$ are used in the decoding algorithm used for decapsulation, $\mathbf{p} \in \mathbb{F}_{2^m}^n$ is a permutation vector and $\mathbf{z} \in \mathbb{F}_2^\ell$ is used in the decapsulation of invalid ciphertexts.

Encapsulation: Given an NTS-KEM public key $\mathrm{pk} = (\mathbf{Q}, \tau, \ell)$, the encapsulation algorithm produces two vectors over $\mathbb{F}_2$ – a random vector $\mathbf{K}$, where $|\mathbf{K}| = \ell$, and the ciphertext $\mathbf{c}^*$ encapsulating $\mathbf{K}$. It uses the following function that acts on n-bit error vectors, and denoted as $\mathrm{Encode}(\mathsf{pk}, \mathbf{e})$, which proceeds as follows.

1. Partition $\mathbf{e}$ as $\mathbf{e} = (\mathbf{e}_a \mid \mathbf{e}_b \mid \mathbf{e}_c)$, where $\mathbf{e}_a \in \mathbb{F}_2^{k-\ell}$, $\mathbf{e}_b \in \mathbb{F}_2^\ell$ and $\mathbf{e}_c \in \mathbb{F}_2^{n-k}$.
2. Compute $\mathbf{k}_e = H_\ell(\mathbf{e}) \in \mathbb{F}_2^\ell$ and construct message vector $\mathbf{m} = (\mathbf{e}_a \mid \mathbf{k}_e) \in \mathbb{F}_2^k$.
3. Perform systematic encoding of $\mathbf{m}$ with $\mathbf{Q}$:

$$\begin{aligned} \mathbf{c} &= (\mathbf{m} \mid \mathbf{m} \cdot \mathbf{Q}) + \mathbf{e} \\ &= (\mathbf{e}_a \mid \mathbf{k}_e \mid (\mathbf{e}_a \mid \mathbf{k}_e) \cdot \mathbf{Q}) + (\mathbf{e}_a \mid \mathbf{e}_b \mid \mathbf{e}_c) \\ &= (\mathbf{0}_a \mid \mathbf{c}_b \mid \mathbf{c}_c)\,, \end{aligned}$$

where $\mathbf{c}_b = \mathbf{k}_e + \mathbf{e}_b$ and $\mathbf{c}_c = (\mathbf{e}_a \mid \mathbf{k}_e) \cdot \mathbf{Q} + \mathbf{e}_c$. Then remove the first $k - \ell$ coordinates (all zero) from $\mathbf{c}$ to output $\mathbf{c}^* = (\mathbf{c}_b \mid \mathbf{c}_c) \in \mathbb{F}_2^{n-k+\ell}$.

NTS-KEM encapsulation is then defined as:

1. Generate uniformly at random an error vector $\mathbf{e} \in \mathbb{F}_2^n$ with $\mathrm{hw}(\mathbf{e}) = \tau$.
2. Compute $\mathbf{k}_e = H_\ell(\mathbf{e}) \in \mathbb{F}_2^\ell$.
3. Output the pair $(\mathbf{K}, \mathbf{c}^*)$ where $\mathbf{K} = H_\ell(\mathbf{k}_e \mid \mathbf{e})$ and $\mathbf{c}^* = \mathrm{Encode}(\mathsf{pk}, \mathbf{e})$.

Decapsulation: [5] The decapsulation of an NTS-KEM ciphertext $\mathbf{c}^* = (\mathbf{c}_b \mid \mathbf{c}_c)$ proceeds as follows.

1. Consider the vector $\mathbf{c} = (\mathbf{0}_a \mid \mathbf{c}_b \mid \mathbf{c}_c) \in \mathbb{F}_2^n$, and apply a decoding algorithm — using the secret parameters $(\mathbf{a}^*, \mathbf{h}^*)$ — to recover a permuted error pattern $\mathbf{e}'$.
2. Compute the error vector $\mathbf{e} = \pi_{\mathbf{p}}(\mathbf{e}')$, partition $\mathbf{e} = (\mathbf{e}_a \mid \mathbf{e}_b \mid \mathbf{e}_c)$, where $\mathbf{e}_a \in \mathbb{F}_2^{k-\ell}$, $\mathbf{e}_b \in \mathbb{F}_2^\ell$ and $\mathbf{e}_c \in \mathbb{F}_2^{n-k}$, and compute $\mathbf{k}_e = \mathbf{c}_b - \mathbf{e}_b$.
3. Compute $\mathbf{c}' = \mathrm{Encode}(\mathsf{pk}, \mathbf{e})$. Verify that $\mathbf{c}' = \mathbf{c}^*$ and $\mathrm{hw}(\mathbf{e}) = \tau$. If yes, return $\mathbf{K} = H_\ell(\mathbf{k}_e \mid \mathbf{e}) \in \mathbb{F}_2^\ell$; otherwise return $H_\ell(\mathbf{z} \mid \mathbf{1}_a \mid \mathbf{c}_b \mid \mathbf{c}_c)$.

3.1 Changes to the Initial NTS-KEM Decapsulation [ACP+19a]

The NIST second round submission for NTS-KEM [ACP+19a] does not perform the re-encoding check in the decapsulation algorithm. Specifically, the evaluation of $\mathrm{Encode}(\mathsf{pk}, \mathbf{e})$ in step 3 of the *Decapsulation* operation above is not performed, and instead it only verifies if $\mathrm{hw}(\mathbf{e}) = \tau$ and $H_\ell(\mathbf{e}) = \mathbf{k}_e$ to identify valid ciphertexts. But this may allow some invalid ciphertexts $\mathbf{c}$ to evade implicit rejection by the decapsulation oracle, leading to a possible attack in the IND-CCA security game of NTS-KEM.

To be specific, the initial IND-CCA security proof for NTS-KEM in the ROM [ACP+19a] failed to account for ciphertexts $\mathbf{c}$ which, when given as input to the decoding algorithm used in NTS-KEM decapsulation, result in an error vector $\mathbf{e}$ such that $\mathrm{hw}(\mathbf{e}) = \tau$ and $H_\ell(\mathbf{e}) = \mathbf{c}_b - \mathbf{e}_b$, but $\mathrm{Encode}(\mathsf{pk}, \mathbf{e}) \neq \mathbf{c}$. Because of the correctness of NTS-KEM (as shown in [ACP+19a]), it is not hard to see that such a ciphertext $\mathbf{c}$ is not the result of any valid NTS-KEM encapsulation.[6] This might lead to a potential attack in the IND-CCA security game of NTS-KEM. Given a challenge ciphertext $\mathbf{c}^* = (\mathbf{c}_b^* \mid \mathbf{c}_c^*)$ (along with a key $\mathbf{K}_b$, see Subsection 2.3 for definitions of security games w.r.t. KEMs), the adversary could possibly construct the above invalid ciphertext $\mathbf{c} = (\mathbf{c}_b^* \mid \mathbf{c}_c)$ by modifying the last $(n-k)$ bits of $\mathbf{c}^*$, such that the decoding algorithm in NTS-KEM decapsulation would recover the error vector $\mathbf{e}^*$ used in the NTS-KEM encapsulation that produced $\mathbf{c}^*$; the attack would then be to query the decapsulation oracle on $\mathbf{c}$ $(\neq \mathbf{c}^*)$ to recover the encapsulated key.

[5] Our suggested routine, as adopted in the updated version of NTS-KEM [ACP+19b].

[6] On the contrary, if there exists an error vector $\mathbf{e}'$ with $\mathrm{hw}(\mathbf{e}') = \tau$ such that $\mathrm{Encode}(\mathsf{pk}, \mathbf{e}') = \mathbf{c}$, then because of NTS-KEM correctness, the decoding algorithm should recover error vector $\mathbf{e}' (\neq \mathbf{e})$ when given $\mathbf{c}$ as input.

At the same time, we stress that the above described attack is just a possibility and is not a *concrete* attack. Because it is quite possible that, by analyzing the decoding algorithm used in NTS-KEM decapsulation, one might show such invalid ciphertexts are computationally hard to generate adversarially.

A re-encoding step during NTS-KEM decapsulation, which is in line with the FO transformations, would entirely resolve this issue by correctly rejecting such invalid ciphertexts. Our proposed changes also perform the hash check $H_\ell(\mathbf{e}) = \mathbf{k}_e$ implicitly because of the following proposition shown in [ACP+19a].

Proposition 1. ([ACP+19a]) Let $\mathbf{c}^* = (\mathbf{c}_b \mid \mathbf{c}_c)$ be a correctly formed ciphertext for NTS-KEM with public key $\mathsf{pk} = (\mathbf{Q}, \tau, \ell)$. Then there exists a unique pair of vectors $((\mathbf{e}_a \mid \mathbf{r}_b), \mathbf{e})$ such that $\mathrm{hw}(\mathbf{e}) = \tau$ and $\mathbf{c}^* = (\mathbf{e}_a \mid \mathbf{r}_b) \cdot [\mathbf{I}_k \mid \mathbf{Q}] + \mathbf{e}$.

If a ciphertext $\mathbf{c}$ is not rejected by the new decapsulation oracle, it means that there is an error vector $\mathbf{e}$ with $\mathrm{hw}(\mathbf{e}) = \tau$ such that $\mathrm{Encode}(\mathsf{pk}, \mathbf{e}) = \mathbf{c}$. From Proposition 1, there then exists a unique pair of vectors $((\mathbf{e}_a \mid \mathbf{r}_b), \mathbf{e})$ w.r.t. $\mathbf{c}$, with $\mathrm{hw}(\mathbf{e}) = \tau$, such that $\mathbf{c} = (\mathbf{e}_a \mid \mathbf{r}_b) \cdot [\mathbf{I}_k \mid \mathbf{Q}] + \mathbf{e}$. It is clear that $\mathbf{r}_b = \mathbf{c}_b - \mathbf{e}_b$, and because of the uniqueness of $\mathbf{r}_b$, we must have $H_\ell(\mathbf{e}) = \mathbf{c}_b - \mathbf{e}_b$ in the evaluation of $\mathrm{Encode}(\mathsf{pk}, \mathbf{e})$. Because of this observation, our changes to NTS-KEM decapsulation also preserve the tightness of the initial IND-CCA security proof for NTS-KEM in the ROM, while fixing the flaw discussed above (refer to [ACP+19b] for more details on the updated ROM proof for NTS-KEM).

4 IND-CCA Security of NTS-KEM in the QROM

In this section, we will be providing a (game-hopping) security proof for NTS-KEM in the QROM, relying on techniques used in [JZC+18,SXY18]. Specifically, we show that NTS-KEM is IND-CCA secure in the QROM, if McEliece is OW secure as a PKE scheme.

Theorem 1. *In the quantum random oracle model, if there exists an adversary $\mathcal{A}$ winning the* IND-CCA *game for NTS-KEM with advantage ε, issuing at most q_D queries to the decapsulation oracle and at most q_H quantum queries to the random oracle $H_\ell(.)$, then there exists an adversary $\hat{\mathcal{B}}$ against the* OW *security of the McEliece PKE scheme with advantage at least $\frac{1}{4}\left(\frac{\varepsilon}{q_H} - \frac{1}{\sqrt{2^{\ell-1}}}\right)^2$, and the running time of $\hat{\mathcal{B}}$ is about that of $\mathcal{A}$.*

Similar to the IND-CCA security proof for NTS-KEM in the classical ROM given in [ACP+19a], we define NTS$^-$, a variant of NTS-KEM, which creates key encapsulations that are McEliece-type encryptions of message vectors of the form $\mathbf{m} = (\mathbf{e_a} \mid \mathbf{r_b})$, where $\mathbf{r_b} \leftarrow\!\!\$\, \mathbb{F}_2^\ell$, and $\mathbf{r_b}$ is considered to be the *encapsulated key* for NTS$^-$. This is in contrast to the original NTS-KEM scheme in which the encapsulations are encryptions of messages of the form $\mathbf{m} = (\mathbf{e_a} \mid \mathbf{k_e})$, where $\mathbf{k_e} = H_\ell(\mathbf{e}) \in \mathbb{F}_2^\ell$, and $\mathbf{K} = H_\ell(\mathbf{k_e} \mid \mathbf{e})$ is the encapsulated key. As will be seen later on, this step is convenient for our proof because we essentially *decouple* the need for random oracles from the NTS$^-$ scheme.

Towards a reduction in the QROM from the IND-CCA security of NTS-KEM to the OW security of the McEliece PKE scheme, as an intermediate step we first note that NTS$^-$ satisfies a non-standard security notion, denoted as *error one-wayness* or EOW security (as introduced in [ACP$^+$19a]). This notion is specific to McEliece-type KEM schemes, e.g., NTS-KEM, which encrypt messages of the form $\mathbf{m} = (\mathbf{e_a} \mid \mathbf{r_b})$ with error vector $\mathbf{e} = (\mathbf{e_a} \mid \mathbf{e_b} \mid \mathbf{e_c})$ during key encapsulation. Roughly speaking, this notion states that it is hard to recover the error vector $\mathbf{e}$ used to generate a given challenge ciphertext $\mathbf{c}$. EOW security for NTS-KEM-like KEMs is defined formally in Figure 4 where the adversary gets the encapsulation of a random key and is required to produce the error vector which led to that particular encapsulation.

$\mathrm{EOW}^{\mathcal{A}}_{\mathsf{KEM}}$

1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$
2 : $(\mathbf{K}, \mathbf{c}^*) \leftarrow \mathsf{Encap}(\mathsf{pk})$
3 : $\mathbf{e}$: error vector used to produce $\mathbf{c}^*$
4 : $\mathbf{e}' \leftarrow \mathcal{A}(1^\lambda, \mathsf{pk}, \mathbf{c}^*)$
5 : **return** $(\mathbf{e}' = \mathbf{e})$

Fig. 4. EOW-security game for KEM.

Now NTS$^-$ is EOW-secure because of the following security reduction shown in [ACP$^+$19a], which does not rely on any random oracles.

Theorem 2. ([ACP$^+$19a]) *If there is a (t, ε)-adversary $\mathcal{B}$ against the* EOW *security of NTS$^-$, then there is a (t, ε)-adversary $\hat{\mathcal{B}}$ against the* OW *security of the McEliece PKE scheme.*

So we can focus on reducing the IND-CCA security of NTS-KEM to the EOW security of NTS$^-$ in the QROM.

Proof. (of Theorem 1) Let $\mathcal{A}$ be an adversary against the IND-CCA game for NTS-KEM with advantage ε, issuing at most q_D queries to the decapsulation oracle and at most q_H queries to the quantum random oracle $H_\ell(.)$.

Consider the games G_0 – G_5 described in Figure 5. Here $\mathsf{pk} = (\mathbf{Q}, \tau, \ell)$ and $\mathsf{sk} = (\mathbf{a}^*, \mathbf{h}^*, \mathbf{p}, \mathbf{z}, \mathsf{pk})$ as described in Section 3 on NTS-KEM key generation. Also $H_0^+ : \{0,1\}^* \to \{0,1\}^\ell, H_1^n : \{0,1\}^n \to \{0,1\}^\ell, H_2^{\ell+n} : \{0,1\}^{\ell+n} \to \{0,1\}^\ell, H_3^n : \{0,1\}^n \to \{0,1\}^\ell$ and $H_4^n : \{0,1\}^n \to \{0,1\}^\ell$ are independent random functions that are used in the evaluation of queries (of varying lengths) w.r.t. the oracles $H_\ell(.)$ and $\mathsf{Decap}(., \mathsf{sk})$ in the games. $\mathrm{Decode}(\mathsf{sk}', \mathbf{c})$ defined over ciphertexts $\mathbf{c} \in \mathbb{F}_2^{n-k+\ell}$ recovers an error vector after applying the decoding algorithm used in NTS-KEM decapsulation [*Decapsulation*, Section 3] and the permutation $\mathbf{p}$ of the secret key.

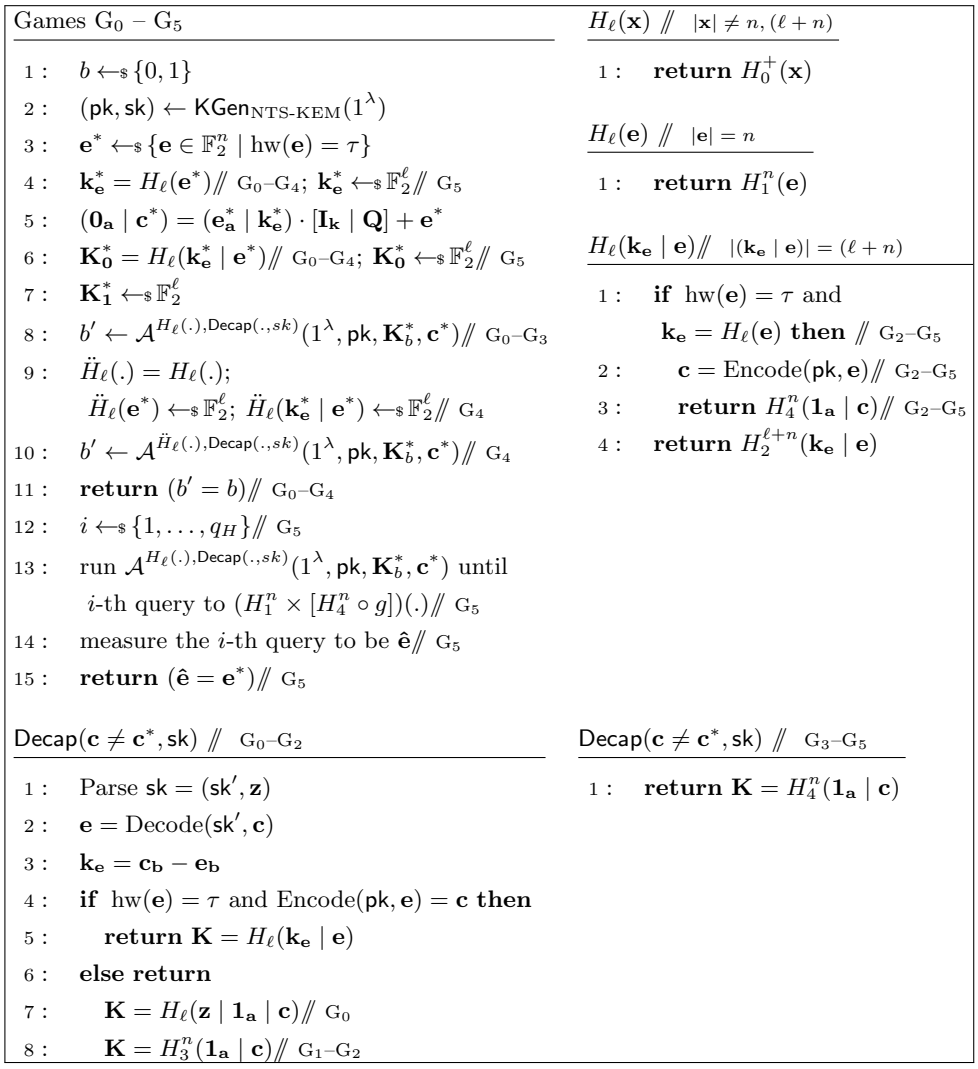

Fig. 5. Games G_0 – G_5 for the proof of Theorem 1.

Game $\mathbf{G}_0$ The game G_0 is exactly the IND-CCA game for NTS-KEM. So,

$$|\Pr[\mathrm{G}_0^{\mathcal{A}} \implies 1] - \frac{1}{2}| = \varepsilon$$

where "$\mathrm{G}_i^{\mathcal{A}} \implies 1$" denotes the event that the game G_i returns 1 w.r.t. the adversary $\mathcal{A}$.

Game $\mathbf{G}_1$ In game G_1, we modify the decapsulation oracle such that $H_3^n(\mathbf{1}_a \mid \mathbf{c})$ is returned instead of $H_\ell(\mathbf{z} \mid \mathbf{1}_a \mid \mathbf{c})$ for an invalid ciphertext $\mathbf{c}$, i.e., pseudo-random decapsulations of invalid ciphertexts are replaced by truly random outputs. We use Lemma 2 to claim that there is a negligible difference between $\mathcal{A}$'s winning probabilities in games G_0 and G_1, i.e.,

$$|\Pr[\mathrm{G}_1^{\mathcal{A}} \implies 1] - \Pr[\mathrm{G}_0^{\mathcal{A}} \implies 1]| \leq q_H \cdot 2^{-\frac{\ell+1}{2}}$$

The proof for this claim follows along similar lines to that of [SXY18, Lemma 2.2], but with modifications to account for the specific way NTS-KEM rejects invalid ciphertexts during decapsulation. To prove our claim, we again consider the following sequence of games for $\mathcal{A}$ based on the random oracles it has access to, that are relevant during the transition from G_0 to G_1.

$\mathbf{G}_0$: The game returns accordingly as $\mathcal{A}^{H_2^{\ell+n}(.),H_2^{\ell+n}(\mathbf{z}|\mathbf{1_a}|.)}(.)$ outputs, where $\mathbf{z} \leftarrow_{\$} \{0,1\}^\ell$ is a part of the secret key sk.

$\mathbf{G}_{0.5}$: The game returns accordingly as $\mathcal{A}^{O^{\ell+n}[\mathbf{z},\mathbf{a},H_2^{\ell+n},H_3^n](.),H_3^n(\mathbf{1_a}|.)}(.)$ outputs, where $O^{\ell+n}[\mathbf{z},\mathbf{a},H_2^{\ell+n},H_3^n](.)$ is a function defined as

$$O^{\ell+n}[\mathbf{z},\mathbf{a},H_2^{\ell+n},H_3^n](\mathbf{z}' \mid \mathbf{c}) = \begin{cases} H_2^{\ell+n}(\mathbf{z}' \mid \mathbf{c}) & \text{if } \mathbf{z}' \neq \mathbf{z} \text{ or } [\mathbf{c}]_a \neq \mathbf{1_a} \\ H_3^n(\mathbf{c}) & \text{otherwise} \end{cases}$$

Here, $[x]_b$ denotes the first b-bits of input x.

$\mathbf{G}_1$: The game returns accordingly as $\mathcal{A}^{H_2^{\ell+n}(.),H_3^n(\mathbf{1_a}|.)}(.)$ outputs.

Note that $\Pr[\mathrm{G}_{0.5}^{\mathcal{A}} \implies 1] = \Pr[\mathrm{G}_0^{\mathcal{A}} \implies 1]$: for $(\ell+n)$-bit queries of the form $(\mathbf{z} \mid \mathbf{1_a} \mid .)$, the function $O^{\ell+n}[\mathbf{z},\mathbf{a},H_2^{\ell+n},H_3^n](.)$ makes sure that we maintain consistency of the oracle evaluations when replacing $H_2^{\ell+n}(\mathbf{z} \mid \mathbf{1_a} \mid .)$ with $H_3^n(\mathbf{1_a} \mid .)$.

We show that $|\Pr[\mathrm{G}_1^{\mathcal{A}} \implies 1] - \Pr[\mathrm{G}_{0.5}^{\mathcal{A}} \implies 1]| \leq q_H \cdot 2^{-\frac{\ell+1}{2}}$ via a reduction to Lemma 2. Consider the algorithm $\mathcal{C}$ that has oracle access to the function $g(.)$ which is either $g_{\mathbf{z}}(.)$ for uniformly random $\mathbf{z} \leftarrow_{\$} \{0,1\}^\ell$ or $g_\perp(.)$, where the functions $g_{\mathbf{z}}(.)$ and $g_\perp(.)$ are as defined in Lemma 2. $\mathcal{C}^{g(.)}$ runs $\mathcal{A}^{\hat{O}^{\ell+n}(.),H_3^n(\mathbf{1_a}|.)}(.)$ where $\mathcal{C}$ simulates the oracles $H_2^{\ell+n}(.)$ and $H_3^n(.)$ using two different $2q_H$-wise independent functions respectively (see Lemma 1), and simulates $\hat{O}^{\ell+n}(.)$ as follows: When $\mathcal{A}$ queries $(\mathbf{z}' \mid \mathbf{c})$ to $\hat{O}^{\ell+n}(.)$, $\mathcal{B}$ queries $\mathbf{z}'$ to $g(.)$ and gets a bit b. If $b = 1$ and $[\mathbf{c}]_a = \mathbf{1_a}$, then $\mathcal{C}$ returns $H_3^n(\mathbf{c})$. Otherwise, $\mathcal{C}$ returns $H_2^{\ell+n}(\mathbf{z}' \mid \mathbf{c})$.

It is clear that if $g(.) = g_{\mathbf{z}}(.)$ for uniformly random $\mathbf{z} \leftarrow_{\$} \{0,1\}^\ell$, $\mathcal{C}$ perfectly simulates $\mathrm{G}_{0.5}$ in $\mathcal{A}$'s view, and similarly if $g(.) = g_\perp(.)$, $\mathcal{C}$ simulates G_1. Thus, we get

$$|\Pr[\mathrm{G}_1^{\mathcal{A}} \implies 1] - \Pr[\mathrm{G}_{0.5}^{\mathcal{A}} \implies 1]| = |\Pr[\mathcal{C}^{g_\perp(.)}(.) \to 1] - \Pr[\mathcal{C}^{g_{\mathbf{z}}(.)}(.) \to 1 \mid \mathbf{z} \leftarrow_{\$} \{0,1\}^\ell]|$$

Since the number of $\mathcal{C}$'s oracle queries to $g(.)$ is the same as the number of $\mathcal{A}$'s queries to $\hat{O}^{\ell+n}(.)$, we can use Lemma 2 to further obtain

$$|\Pr[\mathcal{C}^{g_{\perp}(.)}(.) \to 1] - \Pr[\mathcal{C}^{g_{\mathbf{z}}(.)}(.) \to 1 \mid \mathbf{z} \leftarrow_{\$} \{0,1\}^{\ell}]| \leq q_H \cdot 2^{-\frac{\ell+1}{2}}$$

which proves our claim regarding the adversary $\mathcal{A}$'s winning probabilities in games G_0 and G_1.

Game G_2 In game G_2, the encapsulated keys are derived in a different way: if the $(\ell+n)$-bit input $(\mathbf{k_e} \mid \mathbf{e})$ to $H_\ell(.)$ is of the *correct* form, i.e., $\mathrm{hw}(\mathbf{e}) = \tau$ and $\mathbf{k_e} = H_\ell(\mathbf{e})$, then the output is replaced by $H_4^n(\mathbf{1_a} \mid \mathbf{c})$, where $\mathbf{c} = \mathrm{Encode}(\mathsf{pk}, \mathbf{e})$.

Because NTS-KEM is a *perfectly correct* scheme as shown in [ACP+19a] (see Subsection 2.3 for correctness definition), we note that $\mathrm{Encode}(\mathsf{pk}, .)$ is injective, and thus, $H_4^n(\mathbf{1_a} \mid \mathrm{Encode}(\mathsf{pk}, .))$ returns perfectly random values for distinct inputs of the type $(H_\ell(\mathbf{e}) \mid \mathbf{e})$ with $\mathrm{hw}(\mathbf{e}) = \tau$. As the oracle distributions of $H_\ell(.)$ are equivalent in games G_1 and G_2, we have

$$\Pr[\mathrm{G}_2^{\mathcal{A}} \implies 1] = \Pr[\mathrm{G}_1^{\mathcal{A}} \implies 1]$$

Game G_3 In game G_3, we change the decapsulation oracle such that there is no need for the secret key sk. Specifically, when the adversary $\mathcal{A}$ asks for the decapsulation of a ciphertext $\mathbf{c}$ ($\neq \mathbf{c}^*$, the challenge ciphertext), $H_4^n(\mathbf{1_a} \mid \mathbf{c})$ is returned. Let $\mathbf{e} = \mathrm{Decode}(\mathsf{sk}', \mathbf{c})$ and $\mathbf{k_e} = \mathbf{c_b} - \mathbf{e_b}$. Consider the following two cases:

Case 1: If the checks in NTS-KEM decapsulation – i.e., $\mathrm{hw}(\mathbf{e}) = \tau$ and $\mathrm{Encode}(\mathsf{pk}, \mathbf{e}) = \mathbf{c}$ – are satisfied, then the decapsulation oracles in games G_2 and G_3 return $H_\ell(\mathbf{k}_e \mid \mathbf{e})$ and $H_4^n(\mathbf{1_a} \mid \mathbf{c})$ respectively. In G_2, as discussed in Subsection 3.1, the re-encoding step does an implicit hash check, and hence, we also have $H_\ell(\mathbf{e}) = \mathbf{c}_b - \mathbf{e}_b = \mathbf{k}_e$. Therefore, $H_\ell(\mathbf{k}_e \mid \mathbf{e})$ evaluates to $H_4^n(\mathbf{1_a} \mid \mathrm{Encode}(\mathsf{pk}, \mathbf{e})) = H_4^n(\mathbf{1_a} \mid \mathbf{c})$ in G_2, which is the value returned in G_3 as well.

Case 2: If one of the checks is not satisfied, then the values $H_3^n(\mathbf{1_a} \mid \mathbf{c})$ and $H_4^n(\mathbf{1_a} \mid \mathbf{c})$ are returned in games G_2 and G_3 respectively. In G_2, the function $H_3^n(.)$ is independent of all other random oracles, and thus, the output $H_3^n(\mathbf{1_a} \mid \mathbf{c})$ is uniformly random in $\mathcal{A}$'s view. In G_3, the only way $\mathcal{A}$ gets prior access to the oracle $H_4^n(.)$ is if it already queried $H_\ell(.)$ with an input of the type $(\mathbf{k'_e} \mid \mathbf{e'})$ such that $\mathrm{hw}(\mathbf{e'}) = \tau$ and $\mathbf{k'_e} = H_\ell(\mathbf{e'})$, and got back $H_4^n(\mathbf{1_a} \mid \mathrm{Encode}(\mathsf{pk}, \mathbf{e'}))$. Now it's not hard to see that $\mathrm{Encode}(\mathsf{pk}, \mathbf{e'})$ cannot be equal to $\mathbf{c}$,[7] which implies that the output of the modified decapsulation oracle $H_4^n(\mathbf{1_a} \mid \mathbf{c})$ is a *fresh* random value like $H_3^n(\mathbf{1_a} \mid \mathbf{c})$.

Because the output distributions of the decapsulation oracles in games G_2 and G_3 are the same in both cases, we have

$$\Pr[\mathrm{G}_3^{\mathcal{A}} \implies 1] = \Pr[\mathrm{G}_2^{\mathcal{A}} \implies 1]$$

[7] On the contrary, if $\mathrm{Encode}(\mathsf{pk}, \mathbf{e'}) = \mathbf{c}$, then because of NTS-KEM correctness, we have $\mathrm{Decode}(\mathsf{sk}', \mathbf{c}) = \mathbf{e'} = \mathbf{e}$. This means that the checks $\mathrm{hw}(\mathbf{e}) = \tau$ and $\mathrm{Encode}(\mathsf{pk}, \mathbf{e}) = \mathbf{c}$ are satisfied, a contradiction.

Game G_4 In game G_4, we *reprogram* the random oracle $H_\ell(.)$ on inputs $\mathbf{e}^*$ and $(\mathbf{k}_\mathbf{e}^* \mid \mathbf{e}^*)$ such that they result in fresh uniformly random outputs. To be specific, we replace $H_\ell(.)$ with the function $\ddot{H}_\ell(.)$ where $\ddot{H}_\ell(\mathbf{e}^*) = \mathbf{r}_\mathbf{b}^* \leftarrow_\$ \mathbb{F}_2^\ell$ and $\ddot{H}_\ell(\mathbf{k}_\mathbf{e}^* \mid \mathbf{e}^*) = \dot{\mathbf{K}}_\mathbf{0}^* \leftarrow_\$ \mathbb{F}_\mathbf{2}^\ell$, and $\ddot{H}_\ell(.) = H_\ell(.)$ everywhere else. It is clear that in this game, as we are *masking* the information used to derive the challenge pair $(\mathbf{K}_b^*, \mathbf{c}^*)$ from $\mathcal{A}$'s view, its output is independent of bit b. Therefore,

$$\Pr[\mathrm{G}_4^{\mathcal{A}} \implies 1] = \frac{1}{2}$$

In order to bound the difference in $\mathcal{A}$'s winning probabilities in games G_3 and G_4, we use Lemma 3. Let the function $g(.)$ defined over error vectors $\mathbf{e} \in \mathbb{F}_2^n$ be as follows,

$$g(\mathbf{e}) = \begin{cases} \mathbf{1_a} \mid \mathrm{Encode}(\mathsf{pk}, \mathbf{e}) & \text{if } \mathrm{hw}(\mathbf{e}) = \tau \\ \mathbf{0_a} \mid \mathbf{e_b} \mid \mathbf{e_c} & \text{if } \mathrm{hw}(\mathbf{e}) \neq \tau \end{cases}$$

Looking at game G_3, the oracle query $H_\ell(\mathbf{k}_\mathbf{e}^* \mid \mathbf{e}^*)$ evaluates to $H_4^n(g(\mathbf{e}^*)) = H_4^n \circ g\,(\mathbf{e}^*)$. So we are actually reprogramming the oracles $H_1^n(.)$ and $H_4^n \circ g\,(.)$ at the input $\mathbf{e}^*$.

Define the function $\ddot{H}_4^n \circ g\,(.)$ such that $\ddot{H}_4^n \circ g\,(\mathbf{e}^*) \leftarrow_\$ \mathbb{F}_2^\ell$ and $\ddot{H}_4^n \circ g\,(.) = H_4^n \circ g\,(.)$ everywhere else. Similarly, let $\ddot{H}_1^n(\mathbf{e}^*) \leftarrow_\$ \mathbb{F}_2^\ell$ and $\ddot{H}_1^n(.) = H_1^n(.)$ everywhere else. Now let the oracles $(H_1^n \times [H_4^n \circ g])(.) = (H_1^n(.), H_4^n \circ g\,(.))$[8] and $(\ddot{H}_1^n \times [\ddot{H}_4^n \circ g])(.) = (\ddot{H}_1^n(.), \ddot{H}_4^n \circ g\,(.))$. If we also have a function $\hat{H}_4^n(.)$ such that $\hat{H}_4^n(g(\mathbf{e}^*)) = \perp$ and $\hat{H}_4^n(.) = H_4^n(.)$ everywhere else, then $\hat{H}_4^n(\mathbf{1_a} \mid .)$ is precisely the (unchanged) decapsulation oracle in games G_3 and G_4.

Let $\mathcal{U}^{(H_1^n \times [H_4^n \circ g]), \hat{H}_4^n}$ be an algorithm described in Figure 6 that has quantum access to the oracles $(H_1^n \times [H_4^n \circ g])(.)$ and $\hat{H}_4^n(.)$, and takes an input $(\mathsf{pk}, \mathbf{e}^*, (\mathbf{k}_\mathbf{e}^*, \mathbf{K}_\mathbf{0}^*))$ which is derived in the same way as in games G_3 and G_4; i.e., $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}_{\text{NTS-KEM}}(1^\lambda)$, $\mathbf{e}^* \leftarrow_\$ \{\mathbf{e} \in \mathbb{F}_2^n \mid \mathrm{hw}(\mathbf{e}) = \tau\}, \mathbf{k}_\mathbf{e}^* = H_1^n(\mathbf{e}^*)$ and $\mathbf{K}_\mathbf{0}^* = H_4^n \circ g(\mathbf{e}^*)$, with functions $H_1^n(.), H_4^n(.)$ and $g(.)$ as previously described.

Here the random functions $H_0^+ : \{0,1\}^* \to \{0,1\}^\ell$ and $H_2^{\ell+n} : \{0,1\}^{\ell+n} \to \{0,1\}^\ell$ are independently sampled by the algorithm. Note that, $\mathcal{U}^{(H_1^n \times [H_4^n \circ g]), \hat{H}_4^n}$ on input $(\mathsf{pk}, \mathbf{e}^*, (\mathbf{k}_\mathbf{e}^*, \mathbf{K}_\mathbf{0}^*))$ simulates G_3 in the adversary $\mathcal{A}$'s view, whereas the algorithm $\mathcal{U}^{(\ddot{H}_1^n \times [\ddot{H}_4^n \circ g]), \hat{H}_4^n}$ on the same input $(\mathsf{pk}, \mathbf{e}^*, (\mathbf{k}_\mathbf{e}^*, \mathbf{K}_\mathbf{0}^*))$ simulates G_4. Also $\mathcal{A}$ can have (separate) access to the *internal* oracles $H_1^n(.)$ and $[H_4^n \circ g](.)$ by querying $H_\ell(.)$, which could be simulated by $\mathcal{U}$ by accessing $(H_1^n \times [H_4^n \circ g])(.)$ and ignoring part of the output of the oracle using a trick[9] described in [BZ13,TU16]. Therefore, the number of oracle queries to $(H_1^n \times [H_4^n \circ g])(.)$ is at most q_H.

[8] For error vectors $\mathbf{e} \in \mathbb{F}_2^n$ with $\mathrm{hw}(\mathbf{e}) \neq \tau$, the reason we defined $g(\mathbf{e})$ – even though $\mathcal{A}$ only has access to $H_4^n(\mathbf{1_a} \mid .)$ in games G_3 and G_4 – is to have a consistent domain $(\mathbb{F}_2^n)$ and co-domain $(\mathbb{F}_2^\ell)$ between the oracles $H_1^n(.)$ and $H_4^n \circ g\,(.)$. This would be helpful, for example, when applying Lemma 3 in our setting.

[9] For example, if we want to access $H_1^n(.)$ by making queries to $(H_1^n \times [H_4^n \circ g])(.)$, then we just have to prepare a uniform superposition of all states in the output register corresponding to $H_4^n \circ g(.)$.

$\mathcal{U}^{(H_1^n \times [H_4^n \circ g]), \hat{H}_4^n}(\mathsf{pk}, \mathbf{e}^*, (\mathbf{k}_\mathbf{e}^*, \mathbf{K_0^*}))$

$\mathbf{c}^* = \text{Encode}(\mathsf{pk}, \mathbf{e}^*)$
$\mathbf{K_1^*} \leftarrow_\$ \mathbb{F}_2^\ell$
$b \leftarrow_\$ \{0,1\}$
$b' \leftarrow \mathcal{A}^{H_\ell(.), \mathsf{Decap}(., sk)}(1^\lambda, \mathsf{pk}, \mathbf{K}_b^*, \mathbf{c}^*)$
return $(b' = b)$

$\mathsf{Decap}(\mathbf{c} \neq \mathbf{c}^*, \mathsf{sk})$

return $K = \hat{H}_4^n(\mathbf{1_a} \mid \mathbf{c})$

$H_\ell(\mathbf{x})$ // $|\mathbf{x}| \neq n, (\ell + n)$

return $H_0^+(\mathbf{x})$

$H_\ell(\mathbf{e})$ // $|\mathbf{e}| = n$

return $H_1^n(\mathbf{e})$

$H_\ell(\mathbf{k_e} \mid \mathbf{e})$// $|(\mathbf{k_e} \mid \mathbf{e})| = (\ell + n)$

if $\mathbf{e} = \mathbf{e}^*$ **then**
  if $\mathbf{k_e} = \mathbf{k_e^*}$ **then**
    return $H_4^n \circ g(\mathbf{e})$
  else return $H_2^{\ell+n}(\mathbf{k_e} \mid \mathbf{e})$
elseif $\text{hw}(\mathbf{e}) = \tau$ and $\mathbf{k_e} = H_\ell(\mathbf{e})$ **then**
  return $H_4^n \circ g(\mathbf{e})$
else return $H_2^{\ell+n}(\mathbf{k_e} \mid \mathbf{e})$

Fig. 6. Algorithm $\mathcal{U}^{(H_1^n \times [H_4^n \circ g]), \hat{H}_4^n}$ for the proof of Theorem 1.

Let $\mathcal{V}^{(\ddot{H}_1^n \times [\ddot{H}_4^n \circ g]), \hat{H}_4^n}$ be an algorithm that on input $(\mathsf{pk}, \mathbf{e}^*, (\mathbf{k}_\mathbf{e}^*, \mathbf{K_0^*}))$ does the following: samples $i \leftarrow_\$ \{1, \ldots, q_H\}$, runs $\mathcal{U}^{(\ddot{H}_1^n \times [\ddot{H}_4^n \circ g]), \hat{H}_4^n}$ until the i-th query to $(\ddot{H}_1^n \times [\ddot{H}_4^n \circ g])(.)$ and returns the measurement outcome of the query in the computational basis (if $\mathcal{U}$ makes less than i queries, the algorithm outputs $\perp$).

Game $\mathbf{G}_5$ From the description of G_5, we see that $\Pr[\mathcal{V}^{(\ddot{H}_1^n \times [\ddot{H}_4^n \circ g]), \hat{H}_4^n} \implies \mathbf{e}^*] = \Pr[\mathrm{G}_5^\mathcal{A} \implies 1]$ because, as previously discussed regarding the winning probability in G_4, the oracle $(\ddot{H}_1^n \times [\ddot{H}_4^n \circ g])(.)$ does not reveal any information about $H_\ell(\mathbf{e}^*)$ and $H_\ell(\mathbf{k}_\mathbf{e}^* \mid \mathbf{e}^*)$. So applying Lemma 3 with the setting $\mathcal{O}_1 = (H_1^n \times [H_4^n \circ g])(.), \ddot{\mathcal{O}}_1 = (\ddot{H}_1^n \times [\ddot{H}_4^n \circ g])(.), \mathcal{O}_2 = \hat{H}_4^n(\mathbf{1_a} \mid .), inp = \mathsf{pk}, x = \mathbf{e}^*$ and $y = (\mathbf{k}_\mathbf{e}^*, \mathbf{K_0^*})$, we obtain[10],

$$|\Pr[\mathrm{G}_4^\mathcal{A} \implies 1] - \Pr[\mathrm{G}_3^\mathcal{A} \implies 1]| \leq 2q_H \sqrt{\Pr[\mathrm{G}_5^\mathcal{A} \implies 1]}$$

Finally, we construct an adversary $\mathcal{B}$ against the EOW security of NTS^- (as described in Figure 4) such that its advantage is $\Pr[\mathrm{G}_5^\mathcal{A} \implies 1]$. Given an input $(1^\lambda, \mathsf{pk}, \mathbf{c}^*)$, $\mathcal{B}$ does the following:

- Runs $\mathcal{A}$ as a subroutine in game G_5.
- Uses four different $2q_H$-wise independent functions to *perfectly* simulate the random oracles $H_0^+(.)$, $H_1^n(.), H_4^n(.)$ and $H_2^{l+n}(.)$ respectively in $\mathcal{A}$'s view, as

[10] The original OW2H lemma of [Unr14] would have required $\mathbf{e}^*$ to be sampled uniformly in $\mathbb{F}_2^n$, the domain of $(H_1^n \times [H_4^n \circ g])(.)$. Therefore we use Lemma 3 which generalizes to arbitrary distributions of $\mathbf{e}^*$; in particular, $\mathbf{e}^* \leftarrow_\$ \{\mathbf{e} \in \mathbb{F}_2^n \mid \text{hw}(\mathbf{e}) = \tau\}$.

described in Lemma 1. Also evaluates $H_1^n(.)$ and $[H_4^n \circ g](.)$ at $\mathcal{A}$'s queries using the oracle $(H_1^n \times [H_4^n \circ g])(.)$.
- Answers decapsulation queries using the function $H_4^n(\mathbf{1_a} \mid .)$.
- For $\mathcal{A}$'s challenge query, it samples $\mathbf{K}^* \leftarrow_\$ \mathbb{F}_2^\ell$ and responds with $(\mathbf{K}^*, \mathbf{c}^*)$.
- Samples $i \leftarrow_\$ \{1, \ldots, q_H\}$, measures the i-th query to the oracle $(H_1^n \times [H_4^n \circ g])(.)$ and returns the outcome $\hat{\mathbf{e}}$.

From the above description of $\mathcal{B}$, we note that its EOW advantage against NTS$^-$ is indeed $\Pr[\mathrm{G}_5^{\mathcal{A}} \implies 1]$. Coming to the running times of $\mathcal{A}$ and $\mathcal{B}$, say $t_{\mathcal{A}}$ and $t_{\mathcal{B}}$ respectively, if t_{Enc} denotes the time needed to perform a single Encode($\mathsf{pk}, .$) operation, we have $t_{\mathcal{B}} \approx t_{\mathcal{A}} + (q_H + q_D) \cdot O(q_H) + q_H \cdot t_{\mathrm{Enc}}$, i.e., the overhead is due to the simulation of $H_\ell(.)$ and $\mathsf{Decap}(., sk)$ oracles by $\mathcal{B}$.

By combining the bounds obtained w.r.t. the winning probabilities of $\mathcal{A}$ in each of the previous games and applying the security reduction of Theorem 2 to the EOW adversary $\mathcal{B}$, we obtain an adversary $\hat{\mathcal{B}}$ against the OW security of the McEliece PKE scheme with advantage $\hat{\varepsilon}$, where

$$\hat{\varepsilon} \geq \frac{1}{4}\Big(\frac{\varepsilon}{q_H} - \frac{1}{\sqrt{2^{\ell-1}}}\Big)^2 \tag{1}$$

and the running time of $\hat{\mathcal{B}}$ $(= t_{\mathcal{B}})$ is about that of $\mathcal{A}$.

□

5 Conclusion

In this paper, we analyzed the security of NTS-KEM – a second round PKE/KEM candidate in NIST's PQC standardization project – in the QROM. Specifically, we identified an issue in the IND-CCA security proof of NTS-KEM in the classical ROM and suggested some modifications to the scheme towards fixing it. We later showed that our changes not only preserve the tightness of the intended ROM proof for NTS-KEM but also lead to an IND-CCA security reduction in the QROM. The proposed changes were later adopted by the NTS-KEM team in an update to their second round submission [ACP+19b].

We also note that our QROM reduction can be made tighter by using newer OW2H lemmas of [AHU19] and [BHH+19]. For example, one could consider the improved security reduction of the $\mathrm{U}_m^{\not\perp}$ transform in [BHH+19] to get rid of the factor $1/q_H$ from the term "ε/q_H" in Equation (1). However, the quadratic loss in degree of tightness incurred by our reduction might still be unavoidable in the QROM [JZM19].

References

[ACP+19a] Martin Albrecht, Carlos Cid, Kenneth G. Paterson, CJ Tjhai, and Martin Tomlinson. NTS-KEM: NIST PQC Second Round Submission, 2019. https://nts-kem.io/.

[ACP+19b] Martin Albrecht, Carlos Cid, Kenneth G. Paterson, CJ Tjhai, and Martin Tomlinson. NTS-KEM: NIST PQC Updated Second Round Submission, 2019. `https://nts-kem.io/`.

[AHU19] Andris Ambainis, Mike Hamburg, and Dominique Unruh. Quantum security proofs using semi-classical oracles. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 269–295, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

[BCNP08] Colin Boyd, Yvonne Cliff, Juan Gonzalez Nieto, and Kenneth G. Paterson. Efficient One-Round Key Exchange in the Standard Model. In Yi Mu, Willy Susilo, and Jennifer Seberry, editors, *Information Security and Privacy, 13th Australasian Conference – ACISP 2008*, volume 5107 of *LNCS*, pages 69–83. Springer-Verlag, 2008.

[BDF+11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 41–69, Seoul, South Korea, December 4–8, 2011. Springer, Heidelberg, Germany.

[BHH+19] Nina Bindel, Mike Hamburg, Kathrin Hövelmanns, Andreas Hülsing, and Edoardo Persichetti. Tighter proofs of CCA security in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 61–90, Nuremberg, Germany, December 1–5, 2019. Springer, Heidelberg, Germany.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

[BZ13] Dan Boneh and Mark Zhandry. Secure signatures and chosen ciphertext security in a quantum computing world. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 361–379, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[CS03] Ronald Cramer and Victor Shoup. Design and Analysis of Practical Public-Key Encryption Schemes Secure Against Adaptive Chosen Ciphertext Attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.

[Den03] Alexander W. Dent. A designer's guide to KEMs. In Kenneth G. Paterson, editor, *9th IMA International Conference on Cryptography and Coding*, volume 2898 of *Lecture Notes in Computer Science*, pages 133–151, Cirencester, UK, December 16–18, 2003. Springer, Heidelberg, Germany.

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, January 2013.

[FOPS01] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA – OAEP is secure under the RSA assumption. In J. Kilian, editor, *Advances in Cryptology – CRYPTO 2001*. Springer-Verlag, 2001.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin,

editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 341–371, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany.

[JZC+18] Haodong Jiang, Zhenfeng Zhang, Long Chen, Hong Wang, and Zhi Ma. IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisited. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 96–125, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

[JZM19] Haodong Jiang, Zhenfeng Zhang, and Zhi Ma. On the non-tightness of measurement-based reductions for key encapsulation mechanism in the quantum random oracle model. Cryptology ePrint Archive, Report 2019/494, 2019. `https://eprint.iacr.org/2019/494`.

[McE78] Robert J. McEliece. A Public-Key Cryptosystem Based On Algebraic Coding Theory. *Deep Space Network Progress Report*, 44:114–116, January 1978.

[Nie86] Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. In *Problems of Control and Information Theory 15*, pages 159–166, 1986.

[NIS15] NIST. FIPS PUB 202 Federal Information Processing Standards Publication: SHA-3 Standard: Permutation-Baed Hash and Extendable-Output Functions, August 2015.

[NIS19] NIST. Post-Quantum Cryptography Standardization: Second Round Submissions, January 2019. `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`.

[SXY18] Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. Tightly-secure key-encapsulation mechanism in the quantum random oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 520–551, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[SY17] Fang Song and Aaram Yun. Quantum security of NMAC and related constructions - PRF domain extension against quantum attacks. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 283–309, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[TU16] Ehsan Ebrahimi Targhi and Dominique Unruh. Post-quantum security of the Fujisaki-Okamoto and OAEP transforms. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 192–216, Beijing, China, October 31 – November 3, 2016. Springer, Heidelberg, Germany.

[Unr14] Dominique Unruh. Revocable quantum timed-release encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 129–146, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

[Zha12] Mark Zhandry. Secure identity-based encryption in the quantum random oracle model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances*

in Cryptology – CRYPTO 2012, volume 7417 of *Lecture Notes in Computer Science*, pages 758–775, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.