Security proof of ThreeBears

Mike Hamburg

March 18, 2019

Abstract

This document provides a formal security proof for ThreeBears in the Quantum Random Oracle model (QROM), under the assumption that the Integer Module Learning With Errors (I-MLWE) problem is hard. It considers KEM indistinguishability in the presence of multiple victim keys, multiple challenge ciphertexts per key, against chosen-ciphertext attacks.

Our proof shows explicitly why the unusual features of ThreeBears' hashing modes do not significantly change the security of the construction. These features include a short matrix seed, explicit rejection, and no plaintext-confirmation hash. Furthermore, our proof highlights why some of ThreeBears' features were chosen. We specifically address why the hashing the matrix seed is important for multi-target security, but hashing the whole public key isn't required. We also explain why the IND-CPA mode hashes the seed twice but the IND-CCA mode only hashes it once.

1 Introduction and Related Work

Many of the post-quantum key exchange candidates are based on the on Lyubashevsky-Peikert-Regev [LPR10] RLWE construction. Their security depends primarily on that of RLWE with a certain ring and certain noise parameters. To protect against chosen-ciphertext attacks, they also employ some variant of the Fujisaki-Okamoto transform [FO99]. There are many

possible variants of this transform, depending primarily on:

- Explicit or implicit rejection: does a failed decryption result in a special "failure" return value (usually written $\perp$), or a pseudorandom value?
- Plaintext confirmation: does the ciphertext include a hash of the plaintext?
- Ciphertext hashing: are the message and ciphertext hashed to produce the shared secret, or only the message?
- Domain separation: is the public key included, in whole or in part, when hashing the message?

Each of these options comes with its own trade-offs in performance, security against known attacks, and provable security.

Most proofs of CCA security for practical KEMs use the random oracle model, or in our case the quantum random oracle model (QROM) [BDF+11]. The earlier studies on QROM KEM security, [TU16] and [HHK17], have large gaps in provable security. The IND-CCA advantage is on the order of $\sqrt[4]{q^6\epsilon}$ where ϵ is the IND-CPA advantage against the underlying randomized encryption scheme, and q is the number of quantum random-oracle queries the adversary can make.

More recent studies have reduced the degree of security loss to a square root, and the factor to q^2 or even q, but they come with limitations that prevent them from applying to ThreeBears. Most commonly, existing proofs require either implicit rejection or plaintext confirmation. In particular:

- [SXY18] requires implicit rejection and perfect correctness.
- [JZC+17] requires implicit rejection or plaintext confirmation.
- [XY18] requires implicit rejection.
- [JZM19a] requires plaintext confirmation.
- [JZM19b] requires either implicit rejection or stronger properties from

the underlying encryption scheme, which are usually achieved by plaintext confirmation.

In this paper, we give a proof of IND-CPA and IND-CCA security for ThreeBears, even though ThreeBears uses neither implicit rejection nor plaintext confirmation. While we would like to prove this result for Fujisaki-Okamoto transforms of general public-key encryption schemes, our IND-CCA proof hinges on a particular property of ThreeBears, which may not be shared by systems that use rounding. Specifically, the less-significant bits of each digit in ThreeBears serve the same purpose as a plaintext-confirmation hash.

2 Preliminaries

2.1 Quantum and semi-classical oracles

We will use the one-way-to-hiding (O2H), quantum search and semi-classical oracle results from [AHU18].

Semi-classical oracles A semi-classical oracle $\mathcal{O}_S^{SC}$ for membership in a set S is an oracle which takes a quantum input, and classically measures whether it is in some set S. More formally, $\mathcal{O}_S^{SC}(x)$ quantumly computes whether $x \in S$, storing the result in a zero-initialized auxilliary bit y, and then measures y. Let Find be the event that any such measurement during the computation returns 1. Here S could be given explicitly, or by an efficient quantum algorithm that recognizes it.

Punctured oracles For a quantum-accessible oracle H and a set S, the oracle $H\backslash S$ ("H punctured by S") takes as input a value x, then runs $\mathcal{O}_S^{SC}(x)$, and finally runs $H(x)$ and returns the result. Informally, if the input is in S then the punctured oracle will Find, and otherwise it will return a result independent of H's outputs on S (except insofar as those

outputs are related to other outputs of H). The following lemma formalizes this:

Lemma 1 (Puncturing is effective; [AHU18] Lemma 1)**.** *Let $S \subseteq X$ be random. Let $G, H : X \to Y$ be random functions satisfying $\forall x \notin S.\ G(x) = H(x)$. Let z be a random bitstring. (S, G, H, z may have arbitrary joint distribution.)*

Let A be a quantum oracle algorithm (not necessarily unitary).

Let E be an arbitrary (classical) event.

Then $\Pr[E \wedge \neg\mathsf{Find} : x \leftarrow A^{H\setminus S}(z)] = \Pr[E \wedge \neg\mathsf{Find} : x \leftarrow A^{G\setminus S}(z)]$.

Of course, if we measured the entire input to H then the same theorem would apply, but that would completely ruin the algorithm A whether or not it queries $H(x)$ for $x \in S$. But puncturing only disturbs the adversary's state when it is likely to Find:

Lemma 2 (Semi-classical O2H; [AHU18] Theorem 1, case (4))**.** *Let $S \subseteq X$ be random. Let $G, H : X \to Y$ be random functions satisfying $\forall x \notin S.\ G(x) = H(x)$. Let z be a random bitstring. (S, G, H, z may have arbitrary joint distribution.)*

Let A be an oracle algorithm of query depth d (not necessarily unitary).

Let

$$\begin{aligned} P_{\text{left}} &:= \Pr[b = 1 : b \leftarrow A^H(z)] \\ P_{\text{right}} &:= \Pr[b = 1 \wedge \neg\mathsf{Find} : b \leftarrow A^{H\setminus S}(z)] \\ P_{\text{find}} &:= \Pr[\mathsf{Find} : A^{G\setminus S}(z)] \overset{\text{Lem. 1}}{=} \Pr[\mathsf{Find} : A^{H\setminus S}(z)] \end{aligned}$$

Then

$$|P_{\text{left}} - P_{\text{right}}| \leq 2\sqrt{(d+1) \cdot P_{\text{find}}}$$

We might expect that if the adversary has no information about S, then P_{find} would be at most $q\,|S|\,/\,|X|$. But measuring whether the query is in S disturbs the adversary's state in a way that depends on S, which also gives

information. However, the effect is limited to a factor of 4, as the following lemma shows.

Lemma 3 (Search in semi-classical oracle; [AHU18] Theorem 2)**.** *Let A be any quantum oracle algorithm making at most q queries to a semi-classical oracle with domain X. Let $S \subseteq X$ and $z \in \{0,1\}^*$. (S, z may have arbitrary joint distribution.)*

Let B be an algorithm that on input z chooses $i \xleftarrow{R} \{1,\dots,d\}$; runs $A^{\mathcal{O}_\emptyset^{SC}}(z)$ until (just before) the i-th query; then measures all query input registers in the computational basis and outputs the set T of measurement outcomes.

Then

$$\Pr[\mathsf{Find} : A^{\mathcal{O}_S^{SC}}(z)] \leq 4d \cdot \Pr[S \cap T \neq \varnothing : T \leftarrow B(z)] \tag{1}$$

In some cases, we will only need traditional, fully-quantum O2H. This can be factored into the previous two lemmas, but using a single lemma is cleaner and gives a slightly tighter bound:

Lemma 4 (One-way to hiding; [AHU18] Theorem 3)**.** *Let $S \subseteq X$ be random. Let $G, H : X \to Y$ be random functions satisfying $\forall x \notin S. G(x) = H(x)$. Let z be a random bitstring. (S, G, H, z may have arbitrary joint distribution.)*

Let A be quantum oracle algorithm with query depth d (not necessarily unitary).

Let B^H be an oracle algorithm that on input z does the following: pick $i \xleftarrow{R} \{1,\dots,d\}$, run $A^H(z)$ until (just before) the i-th query, measure all query input registers in the computational basis, output the set T of measurement outcomes.

Let

$$\begin{aligned} P_{\text{left}} &:= \Pr[b = 1 : b \leftarrow A^H(z)] \\ P_{\text{right}} &:= \Pr[b = 1 : b \leftarrow A^G(z)] \\ P_{\text{guess}} &:= \Pr[S \cap T \neq \varnothing : T \leftarrow B^H(z)] \end{aligned}$$

Then

$$|P_{\text{left}} - P_{\text{right}}| \leq 2d\sqrt{P_{\text{guess}}} \qquad \text{and} \qquad \left|\sqrt{P_{\text{left}}} - \sqrt{P_{\text{right}}}\right| \leq 2d\sqrt{P_{\text{guess}}}$$

The same result holds with B^G instead of B^H in the definition of P_{guess}.

We will use three corollaries to this theorem. The first shows that unstructured search algorithms (such as Grover's algorithm) have performance bounded by their query depth.

Corollary 1 (Unstructured quantum search; tightening of [AHU18] Lemma 2)**.** *Let H be a random function from $X \to \{0,1\}$, drawn from a distribution such that $\Pr[H(x) = 1] \leq \lambda$ for all x. Let A be a q-query adversary with query depth d. Then $\Pr[H(x) = 1 : b \leftarrow A^H()] \leq 4(d+1)(q+1)\lambda$.*

Proof. Let B^H be the algorithm which runs $x \leftarrow A^H()$ and then measures $H(x)$. It makes $q+1$ queries at depth $d+1$. Applying Lemma 4 to B^H vs B^0, where in the latter case the oracle always returns 0, gives

$$\left|\sqrt{P_{\text{left}}} - \sqrt{P_{\text{right}}}\right| \leq 2(d+1)\sqrt{P_{\text{guess}}}$$

where $P_{\text{right}} = 0$ and $P_{\text{guess}} \leq \lambda(q+1)/(d+1)$. Substituting and squaring both sides gives the claimed result. □

A second corollary is that a random oracle is suitable as a pseudorandom generator. That is, hashing short random values is as secure as choosing longer random values, so long as the short ones aren't short enough to fall to a Grover attack.

Corollary 2 (Replacing random oracle results with random values)**.** *Let H be a random oracle from $X \to Y$, which for each $x \in X$ returns a uniformly, independently random value y from some distribution D_Y on Y. Let x be a random variable, drawn from a distribution D_X such that $\Pr[x = x_0] \leq \lambda$ for all $x_0 \in X$. Let k be an integer greater than the size of the support of D_X. Let $x_1, \ldots, x_k \xleftarrow{u} D$ denote indicate drawing k samples from D, and rejecting until all of them are unique. Let A be a q-query algorithm with query depth d.*

$$P_{\text{left}} := \Pr[b = 1 : x_1, \ldots, x_k \xleftarrow{u} D_X; b \leftarrow A^H(H(x_1), \ldots, H(x_k))]$$
$$P_{\text{right}} := \Pr[b = 1 : y_1, \ldots, y_k \leftarrow D_Y; b \leftarrow A^H(y_1, \ldots, y_k)]$$

Then

$$|P_{\text{left}} - P_{\text{right}}| \leq 2\sqrt{qdk\lambda}$$

Proof. The right case is exactly equivalent to changing the return value of H to a different uniform random value on the set $\{x_1, \ldots, x_k\}$. Each x is in this set with probability at most $k\lambda$. Then the result follows directly from Lemma 4. □

As a third corollary, we show that when replacing a 2-input oracle $H(x, y)$, it is good enough if the distribution matches for most x.

Corollary 3 (Curried oracles)**.** *Let G resp. H be random oracles from $X \times Y \to Z$, where X and Y are finite, such that for each x, the function $G(x\cdot)$ resp. $H(x, \cdot)$ are independent random samples from some distribution D resp. E of functions from $Y \to Z$. Suppose that the total variation distance between D and E is at most λ. Then for all algorithms A that make q queries at depth d and return a bit b,*

$$\left|\Pr[b = 1 : b \leftarrow A^H()] - \Pr[b = 1 : b \leftarrow A^G()]\right| \leq 2\sqrt{qd\lambda}$$

Proof. This is an information-theoretic argument, so we don't need to be efficient. Factor D and E as

$$D = \begin{cases} D_0 & \text{with probability } 1 - \lambda \\ D_1 & \text{with probability } \lambda \end{cases}, \quad E = \begin{cases} D_0 & \text{with probability } 1 - \lambda \\ E_1 & \text{with probability } \lambda \end{cases}$$

Let G' resp. H' be jointly distributed oracles from $X \to (Y \to Z)$, which (independently for each input $x \in X$) return the same random element sampled according to D_0 with probability $1 - \lambda$, and possibly-different elements sampled according to D_1 resp. E_1 with probability λ. Define the uncurrying function $\text{unc}^F(x, y) = F(x)(y)$. Then $A^{\text{unc}^{G'}}$ resp. $A^{\text{unc}^{H'}}$ behave identically to A^G resp. A^H. Furthermore for each x, $\Pr[G'(x) \neq H'(x)] \leq \lambda$. The result then follows from Lemma 4. □

2.2 Indifferentiable hashing

Two random oracles G and H are said to be δ-indifferentiable [MRH04] if there are efficient algorithms $\hat{G}^H$ and $\hat{H}^G$, which efficiently approximate G and H respectively with access to the other, such that an adversary cannot tell the difference between $(\hat{G}^H, H)$ and $(G, \hat{H}^G)$ with probability greater than δ. In the random oracle model, a system which is secure with oracle H is necessarily secure (with additive loss at most δ) when instantiated with $\hat{H}^G$ and vice-versa. Note that a tuple of oracles $(H_1, H_2, \ldots)$ may be treated the same as a single random oracle H with a larger input, where $H_1(x) = H(1, x)$ etc.

3 Security model

This section covers the security model of a public-key encryption system (KeyGen, Enc, Dec) or a key-encapsulation mechanism ($\text{KeyGen}_{\text{KEM}}$, Encaps, Decaps). The KeyGen and Enc/Encaps algorithms are randomized, which we represent through them taking an extra "coins" argument from a space $\mathcal{C}_{\text{Keygen}}$ etc. The Enc and Dec algorithms encrypt or decrypt a message $m \in \mathcal{M}$, and the KEM algorithms share a shared secret key $\text{ss} \in X_{\text{ss}}$.

We aim to model an adversary who has access to a large number k of public keys and c challenge ciphertexts per key. We could give the adversary these keys and ciphertexts up front, but if k and c are both large (say 2^{64}), then kc could be truly enormous. So instead we give the adversary one oracle to generate public keys and another to generate challenge ciphertexts for a given key. We limit the adversary to c ciphertexts per key and c_{tot} total.

The IND-CPA (Indistiguishability under Chosen-Plaintext Attack) game for encryption is shown in Algorithm 1. In the QROM, all algorithms are relative to the random oracle(s), and the adversary has quantum access to those oracles. We use indistinguishability, rather than one-way-ness (OW-passive), for two reasons. First, it fits better with LWE. Decision-LWE gives IND-CPA security, but the harder problem (search-LWE) doesn't give the

weaker security notion (OW-passive). Second, CCA reductions based on IND-CPA seem tighter than those based on OW-passive.

We could reduce to a weaker IND-KPA (*Known*-Plaintext Attack) model, but the message is just added to the masking elements anyway, so this wouldn't gain anything. We use IND-CPA instead because it simplifies choosing messages from random oracles. We could also define an IND-CCA (Chosen-*Ciphertext* Attack) game for encryption, but we don't need it for this paper since we only want to prove IND-CCA for KEMs.

Algorithm 1: CPA security game for public-key encryption

Game IND-CPA($\mathcal{A}$) **is**
  $b_{\text{chal}} \xleftarrow{R} \{0,1\}$;
  $k \leftarrow 0$;
  Classical oracle ChalKeygen() **is**
    $k \leftarrow k+1$;
    coins $\leftarrow \mathcal{C}_{\text{Keygen}}$;
    $(\text{pk}_k, \text{sk}_k) \leftarrow \text{Keygen}(\text{coins})$;
    return pk_k;
  end
  Classical oracle ChalEnc(i, m_1) **is**
    if $i > k$ **then return** $\perp$;
    $m_0 \xleftarrow{R} \mathcal{M}$;
    coins $\xleftarrow{R} \mathcal{C}_{\text{Enc}}$;
    return $\text{Enc}(\text{pk}_i, m_{b_{\text{chal}}}, \text{coins})$;
  end
  $b_{\text{guess}} \leftarrow \mathcal{A}^{\text{ChalKeygen,ChalEnc}}()$;
  $\mathcal{A}$ wins if and only if $b_{\text{guess}} = b_{\text{chal}}$.
end

For KEMs, our IND-CCA (Indistiguishability under Chosen-Ciphertext Attack) game is shown in Algorithm 2. This is like the IND-CPA game for encryption, except that the adversary doesn't have access to the plaintext.

Instead, it learns a candidate shared secret, which in the case of THREEBEARS is the hash of the plaintext. Furthermore, since are modeling a chosen-ciphertext attack, the adversary gets access to a decryption oracle ChalDec. That oracle isn't allowed to decrypt challenge capsules, but instead returns the failure symbol $\perp$. The IND-CPA security game for KEMs is the same, except that the adversary doesn't have access to ChalDec.

Algorithm 2: IND-CCA security game for KEMs.

Game IND-CCA($\mathcal{A}$) **is**
  $b_{\text{chal}} \xleftarrow{R} \{0,1\}$;
  $k \leftarrow 0$;
  $C \leftarrow \emptyset$;
  Classical oracle ChalKeygen() **is**
    $k \leftarrow k+1$;
    $(\text{pk}_k, \text{sk}_k) \leftarrow \text{Keygen}_{\text{KEM}}()$;
    return pk_k;
  end
  Classical oracle ChalEnc(i) **is**
    if $i > k$ **then return** $\perp$;
    $(\text{ct}, \text{ss}_1) \leftarrow \text{Encaps}(\text{pk}_i)$;
    $\text{ss}_0 \xleftarrow{R} X_{\text{ss}}$;
    $C \leftarrow C \cup \{(i, \text{ct})\}$;
    return (ct, ss_b);
  end
  Classical oracle ChalDec(i, ct) **is**
    if $i > k$ **or** $(i, \text{ct}) \in C$ **then return** $\perp$;
    return $\text{Decaps}(\text{sk}_i, \text{ct})$;
  end
  $b_{\text{guess}} \leftarrow \mathcal{A}^{\text{ChalKeygen,ChalEnc,ChalDec}}()$;
  $\mathcal{A}$ wins if and only if $b_{\text{guess}} = b_{\text{chal}}$.
end

For an IND-CPA adversary $\mathcal{A}$, define its (encryption or KEM) IND-CPA

advantage as

$$\begin{aligned}\mathrm{Adv}_{\mathrm{IND-CPA}}(\mathcal{A}) &:= |2 \cdot \Pr[\mathcal{A} \text{ wins the IND-CPA game}] - 1| \\ &= |\Pr[1 \leftarrow \mathcal{A} : b_{\mathrm{chal}} = 1] - \Pr[1 \leftarrow \mathcal{A} : b_{\mathrm{chal}} = 0]|\end{aligned}$$

and likewise for CCA-advantage.

4 Simplifications

In this section, we show that changing two features of ThreeBears, its key generation from the seed, and its shorter-than-usual matrix seed, doesn't have any serious security consequences. This will simplify the rest of the security proof.

4.1 Key generation without the seed

We first change ThreeBears to a variant, ThreeBears$_0$, which chooses the coins for private key generation uniformly at random instead of expanding from a 320-bit seed.

By Corollary 2, if the adversary $\mathcal{A}$ makes q random oracle queries at depth d, then

$$\left| \begin{array}{l} \mathrm{Adv}_{\mathrm{IND-CCA}}(\mathcal{A} : \text{ThreeBears}_0) \\ \quad -\mathrm{Adv}_{\mathrm{IND-CCA}}(\mathcal{A} : \text{ThreeBears}) \end{array} \right| \leq 4\sqrt{qdk/2^{320}} + k^2/2^{320}$$

and the same for IND-CPA. With $k \leq 2^{64}$, this is almost identical to the bound for a Grover attack on a 256-bit secret.

4.2 No collisions in the matrix seed

ThreeBears public keys include a 192-bit matrix seed, and the collision probability of such seeds over a 2^{64}-key attack is less than $\binom{k}{2}/s < 2^{-65}$, where $s = 2^{192}$ is the seed space. This is a tiny probability, but perhaps

not negligible. In this paper we will play a modified game where the matrix seeds are chosen to be distinct.

Formally, let Unique be the event that all keys generated during the game have distinct `matrixSeed`. We define the IND-CPA-U advantage of an adversary $\mathcal{A}$ as the conditional advantage:

$$\begin{aligned}\mathrm{Adv}_{\mathrm{IND-CPA-U}}(\mathcal{A}) &:= |2 \cdot \Pr[\mathcal{A} \text{ wins the IND-CPA game} : \mathsf{Unique}] - 1| \\ &= \left| \begin{array}{l} \Pr[1 \leftarrow \mathcal{A} : b_{\mathrm{chal}} = 1, \mathsf{Unique}] \\ \quad - \Pr[1 \leftarrow \mathcal{A} : b_{\mathrm{chal}} = 0, \mathsf{Unique}] \end{array} \right|\end{aligned}$$

In ThreeBears$_0$ the seed is uniformly random, and not even generated by an oracle, so this is equivalent to the advantage in an "IND-CPA-U game" where the challenger samples the `matrixSeed` uniformly from all 192-bit strings that haven't been sampled yet. Define IND-CCA-U analogously. Note that the adversary has the same description in both cases. It's only the challenger's behavior that differs.

The unique-key game cannot increase the adversary's advantage by much, as the following lemma shows:

Lemma 5. *If $\mathcal{A}$ is an IND-CPA resp. IND-CCA adversary against* ThreeBears$_0$ *with advantage ϵ, and suppose that it requests at most k public keys. Then there is an IND-CPA resp. IND-CCA adversary $\mathcal{B}$ against* ThreeBears$_0$*, running in about the same time as $\mathcal{A}$, that has IND-CPA-U resp. IND-CCA-U advantage*

$$\epsilon_u \geq \max\left(\epsilon - \binom{k}{2}/s, \quad \epsilon/2 - \binom{k}{3}/s^2, \quad \epsilon/3 - \binom{k}{4}/s^3, \quad \ldots\right)$$

so that

$$\epsilon \leq \min\left(\epsilon_u + \frac{k^2}{2s^1}, \quad 2\epsilon_u + \frac{k^3}{3s^2}, \quad 3\epsilon_u + \frac{k^4}{4s^3}, \quad \ldots\right)$$

Proof. Deferred to Appendix C. □

In particular, with $k \leq 2^{64}$ and $s = 2^{192}$, we will have

$$\epsilon \leq \min\left(\epsilon_u + 2^{-65}, \quad 2\epsilon_u + 2^{-193}\right)$$

so that the shorter seed costs at most about 1-2 bits of security.

4.3 As a simple PKE algorithm

We further define ThreeBears$_r$ the public-key encryption algorithm underlying ThreeBears$_0$, which encrypts a message m with random coins, and doesn't hash m to produce a shared secret. The system ThreeBears$_r$ doesn't call the random oracle except to expand the matrix seed.

5 IND-CPA-U security proof

So far we have reduced the IND-CPA security of ThreeBears to IND-CPA-U security of ThreeBears$_0$. We next show that the underlying PKE algorithm ThreeBears$_r$ is IND-CPA-U secure. We start with a definition of the MLWE problem.

Definition (MLWE)**.** *Let R be a finite ring. Let χ be a probability distribution over R. Let d_1 and d_2 be positive integers. The $(R, \chi, d_1 \times d_2)$-MLWE problem is to distinguish the MLWE distribution*

$$\mathcal{D}_1 := \{(M, Ma + e) : M \xleftarrow{R} R^{d_1 \times d_2}, a \leftarrow \chi^{d_1}, e \leftarrow \chi^{d_2}\}$$

from the uniform distribution

$$\mathcal{D}_0 := \{(M, r) : M \xleftarrow{R} R^{d_1 \times d_2}, r \xleftarrow{R} R^{d_2}\}$$

A (possibly randomized) algorithm $\mathcal{A} : R^{(d_1+1)\times d_2} \to \{0, 1\}$ has MLWE advantage:

$$\mathrm{Adv}_{\mathrm{MLWE}}(\mathcal{A}) := |\Pr[\mathcal{A}(X) \to 1 : X \leftarrow D_1] - \Pr[\mathcal{A}(X) \to 1 : X \leftarrow D_0]|$$

The I-MLWE problem is just the MLWE problem with $R = \mathbb{Z}/N$ for $N \approx x^D$, and χ as a distribution which samples elements $e \in R$ with small digits in radix x. In the case of ThreeBears, χ is the `noise` function. Note that the definition is the same with or without an (invertible) clarifier, because if M is uniformly distributed in $R^{d_1 \times d_2}$ then so is clar$\cdot M$, and vice-versa.

The next lemma shows that if the $(d+1) \times d$ I-MLWE problem is hard (with $\chi =$ `noise`), then ThreeBears$_r$ is IND-CPA-U secure.

Lemma 6. *Let $\mathcal{A}$ be an IND-CPA adversary against* ThreeBears$_r$*, which instantiates k public keys and at most c challenge ciphertexts per key. Then*

$$\begin{aligned}&\mathrm{Adv}_{\mathrm{IND-CPA-U}}(\mathcal{A} : \textsc{ThreeBears}_r) \\ &\qquad \leq 2k(c+1) \cdot (\mathrm{Adv}_{((\mathrm{d}+1)\times\mathrm{d})-\mathrm{I-MLWE}}(\mathcal{B}) + 2^{-1555})\end{aligned}$$

Proof. A standard hybrid argument, deferred to Appendix B. □

Why bother addressing multi-target attacks if the proof loses a factor of $2k(c+1)$? The answer is twofold. First, while we expect that the attacker will gain some advantage from multi-target attacks, we don't expect a full factor of $2k(c+1)$. So the rest of the paper can model the impact of multi-target attacks on the Fujisaki-Okamoto transform. We could invent an "I-MLWE($d\times(d+1); k\times c$)" problem for the reduction, but the definition would be almost the same as the security of ThreeBears$_r$ anyway. Second, for IND-CCA security, the IND-CPA advantage will end up inside a square root. Addressing multi-target attacks explicitly puts the $kc+k$ factor inside the square root instead of outside.

5.1 As a KEM

Theorem 1. *Let $\mathcal{A}$ be an IND-CPA adversary against* ThreeBears$_0$*. Suppose $\mathcal{A}$ queries the random oracles at most q times with query depth at most d, and uses at most k public keys, c challenge ciphertexts per key, and c_{tot} challenge ciphertexts total. Then there exists an IND-CPA adversary against* ThreeBears$_r$ *with the same (k, c, c_{tot}) such that*

$$\begin{aligned}\mathrm{Adv}_{\mathrm{IND-CPA-U}}(\mathcal{A} : \textsc{ThreeBears}_0) \quad &\leq \quad 2\mathrm{Adv}_{\mathrm{IND-CPA-U}}(\mathcal{A} : \textsc{ThreeBears}_r) \\ &+ \quad c \cdot c_{\mathrm{tot}} / |M| + 6\sqrt{dqc / |M|}\end{aligned}$$

Proof. We prove security by a series of games. During those games, we will say that if certain classical events occur, the game is a draw, which is a third outcome distinct from the adversary winning or losing. Let

$$w_i \quad := \quad \Pr[\mathcal{A} \text{ wins game } i] + \frac{1}{2}\Pr[\mathcal{A} \text{ draws game } i]$$

Game 0 (IND-CPA)**.** *This is the IND-CCA-U game against* ThreeBears$_0$.

Game 1 (Seeds can't collide)**.** *Game 1 is the same as Game 0, except that if two challenge seeds collide, then the game is a draw.*

We have $|w_1 - w_0| \leq c \cdot c_{\text{tot}}/(4\,|M|)$

Game 2 (Random m and coins)**.** *Game 2 is like Game 1, except that the messages and encryption coins are chosen at random instead of being expanded from the seed.*

By Corollary 2, $|w_2 - w_1| \leq 2\sqrt{dqc/\,|M|}$

Game 3 (Encryptions of random messages)**.** *Game 3 is like Game 2, except that each ciphertext encrypts a random message instead of the challenge message.*

We have $|w_3 - w_2| \leq \text{Adv}_{\text{IND-CPA-U}}(\mathcal{A} : \text{ThreeBears}_r)$.

Game 4 (Messages can't collide)**.** *Game 4 is the same as Game 3, except that if two challenge messages collide, then the game is a draw.*

We have

$$|w_4 - w_3| \leq c \cdot c_{\text{tot}}/(4\,|M|)$$

But now if $b = 1$ the shared secrets are hashes of random distinct messages used nowhere else, and if $b = 0$ they are uniformly random values. By Corollary 2, we have

$$\left| w_4 - \frac{1}{2} \right| \leq \sqrt{dqc/\,|M|}$$

Summing up and doubling these differences give the claimed advantage. □

Recall that ThreeBears' IND-CPA mode differs from its IND-CCA mode partly in that for IND-CPA, the coins and message are the hash of a seed, but for IND-CCA the message is the seed. The reason for this split is seen in the above proof. If the coins were the hash of the message, we would have a circular-security problem, and we would need a looser and more complex

reduction involving semiclassical oracles. We could not avoid this problem for IND-CCA, because everything must be derived from the message so that the recipient can check the encryption.

6 IND-CCA security proof

This section shows a reduction from k-key IND-CCA-U security of ThreeBears$_0$ to IND-CPA-U security of ThreeBears$_r$, where cSHAKE is modeled as a quantum-accessible random oracle. In the simulation we will additionally assume a quantum-accessible ideal cipher, but only for simplicity. The same effect can be achieved with a standard-model cipher or at the cost of decreased performance with many challenge ciphertexts.

We begin with some notation.

6.1 High and low bits

Let $N = 2^{3120} - 2^{1560} - 1$ be the ThreeBears modulus. For a number $y \in \mathbb{Z}/N$, let $\text{high}_4(y)$ be the concatenation of the most-significant 4 bits of each 10-bit digit of y, and let $\text{low}_6(y)$ the concatenation of the least-significant 6-bits of each 10-bit digit of y. Extend this notation to vectors of length d by concatenation, meaning that

$$\text{high}_4((y_0, y_1, \ldots, y_{d-1})) := \text{high}_4(y_0) \mid\mid \text{high}_4(y_1) \mid\mid \ \ldots \ \mid\mid \text{high}_4(y_{d-1})$$

and likewise for low_6.

For brevity, for a public key pk, let

$$H_b(\text{pk}, m) := [\![\texttt{noise}_2(\text{matrixSeed}||m||\text{iv}, i)]\!]_{i=0}^{d-1}$$

Let $H_e(\text{pk}_m)$ be the same, except with $i = d$ to $2d - 1$. Note that iv is the empty string for all recommended instances of ThreeBears.

6.2 High-bit-injective keys

Fix H_b. For a public key pk, let M be the $d \times d$ matrix derived by expanding its seed. For a plaintext m, let $\vec{B}$ be the ciphertext component $M \cdot H_b(\text{pk}, m) + H_e(\text{pk}, m)$. Call pk "high-bit-injective" if, for all plaintexts $(m_0, \text{iv}_1) \neq (m_1, \text{iv}_2)$ producing secrets $\vec{b}_0$ resp $\vec{b}_1$, and for all possible noise values $(\vec{e}_0, \vec{e}_1)$ (not just those that result from $H_e(\text{pk}, m)$), we have

$$\text{high}_4(M \cdot H_b(\text{pk}, m_0) + \vec{e}_0) \neq \text{high}_4(M \cdot H_b(\text{pk}, m_1) + \vec{e}_1)$$

High-injectivity is a property of the matrix M and $H_b(\text{pk}, \cdot)$, but not $H_e(\text{pk}, \cdot)$.

Lemma 7. *Denote by ϵ_{hi} the probability that a* ThreeBears$_0$ *public key fails to be high-bit-injective. Then for recommended* ThreeBears *instances, $\epsilon_{\text{hi}} \leq 2^{-409}$.*

Proof. Deferred to Appendix A. □

Most CCA security proofs use a plaintext-confirmation tag. The actual security benefit of this tag is questionable, but in the proof it is used with a backdoored oracle to leak the plaintext to the simulator. We show that the same can be accomplished using the low bits of the message, which are influenced almost directly by H_e.

Lemma 8. *For a certain uniformly random oracle family F, there is a pair of efficient hash functions $G_e(\text{pk}, m), G_t(\text{pk}, m)$ such that if* pk *is high-bit-injective, then:*

- *(F, H_b, H_e) is 0-indifferentiable from (F, H_b, G_e, G_t).*
- *G_t is a uniformly random oracle which returns a 256-bit value.*
- *$G_t(\text{pk}, m)$ can be calculated from only the ciphertext component $B = Mb + e$, except with probability at most $\epsilon_t < 2^{-319}$ independently per message.*

Proof. Deferred to Appendix D □

We note that a similar construction should work for other (Ring, Module and Plain) LWE schemes, but not necessarily for LWR.

For the proof we will actually use the oracle

$$G_t(\mathrm{pk}, m) := m \oplus G_m(\mathrm{high}_4(M \cdot H_b(\mathrm{pk}, m) + \vec{H}_e(\mathrm{pk}, m)))$$

where G_m is another random oracle. If pk is high-bit-injective, then this oracle is also uniformly random (but not indifferentiable from G_t).

6.3 Main theorem

We are now ready for the main IND-CCA security theorem.

Theorem 2. *Let $\mathcal{A}$ be an IND-CCA adversary against* ThreeBears$_0$. *Suppose $\mathcal{A}$ makes at most q queries to the challenge oracles and cSHAKE (modeled as a quantum-accessible random oracle) at depth d; uses at most k keys, c challenge ciphertexts per key and c_{tot} challenge ciphertexts total; and asks at most q_{dec} decryption queries. Let $\delta_{\max}$ be the expected maximum failure probability encountered in k keys. Then there is an adversary $\mathcal{B}$ using only slightly more resources than $\mathcal{A}$, such that*

$$\begin{aligned}
&\mathrm{Adv}_{\mathrm{IND-CCA-U}}(\mathcal{A} : \textsc{ThreeBears}_0) \\
&\quad \le 2\sqrt{(d+1) \cdot (\mathrm{Adv}_{\mathrm{IND-CPA-U}}(\mathcal{B} : \textsc{ThreeBears}_r) + 8cq/\left|\mathcal{M}\right|)} \\
&\quad + 8dq(\delta_{\max} + \epsilon_t) \\
&\quad + \frac{c \cdot c_{\mathrm{tot}}}{|\mathcal{M}|} + \frac{2c \cdot q_{\mathrm{dec}}}{|\mathcal{M}| - c} + 4\sqrt{dq\epsilon_{\mathrm{hi}}} + k\epsilon_{\mathrm{hi}}
\end{aligned}$$

The main differences in runtime between $\mathcal{B}$ and $\mathcal{A}$ are that $\mathcal{B}$ includes most of the IND-CCA challenger; and that $\mathcal{B}$ replaces calls to H_e with a sampling algorithm, which requires solving a 256×256 system of linear equations over F_2.

For a typical quantum attack we should have $q \gg c_{\mathrm{tot}}, q \gg q_{\mathrm{dec}}, q > d \gg 1, |\mathcal{M}| \gg c$ and $(\epsilon_{\mathrm{hi}}, \epsilon_t)$ negligible, so that

$$\begin{aligned}
&\mathrm{Adv}_{\mathrm{IND-CCA-U}}(\mathcal{A} : \textsc{ThreeBears}_0) \\
&\quad \lessapprox 2\sqrt{d \cdot \mathrm{Adv}_{\mathrm{IND-CPA-U}}(\mathcal{B} : \textsc{ThreeBears}_r)} + 4\sqrt{2dqc/\left|\mathcal{M}\right|} + 4\sqrt{dq\delta_{\max}}
\end{aligned}$$

In turn $\mathrm{Adv}_{\mathrm{IND-CCA}}(\mathcal{A} : \textsc{ThreeBears})$ should be at most about twice this much.

6.4 Proof of Theorem 2

We now prove security by a series of games. During those games, we will say that if certain classical events occur, the game is a draw, which is a third outcome distinct from the adversary winning or losing. Let

$$w_i \;:=\; \Pr[\mathcal{A} \text{ wins game } i] + \frac{1}{2}\Pr[\mathcal{A} \text{ draws game } i]$$

Note that this definition makes sense even if draws aren't efficiently detectable.

Game 0 (IND-CCA-U)**.** *This is the IND-CCA-U security game against* $\textsc{ThreeBears}_0$.

Game 1 (High-bit-injective public keys)**.** *This is the same as Game 0, except that the game is a draw if not all challenge keys are high-bit-injective.*

We have $|w_1 - w_0| \leq k\epsilon_{\mathrm{hi}}/2$.

Game 2 (Messages can't collide)**.** *This is the same as Game 1, except that the challenge plaintexts are chosen always to be unique. For clarity, the ith challenge plaintext to a public key* pk *is* $\pi_{\mathtt{matrixSeed}(\mathrm{pk})}(i)$*, where* π *is an ideal cipher known only to the simulator.*

This is necessary, because the adversary wins with high probability whenever challenge plaintexts collide, simply by testing whether the shared secrets collide. We have $|w_2 - w_1| \leq \frac{1}{2}c \cdot c_{\mathrm{tot}}/\,|\mathcal{M}|$.

Game 3 (No prescient queries)**.** *This is the same as Game 2, except that the decapsulation oracle returns the failure code* $\perp$ *whenever the decapsulated plaintext* m *is a possible challenge message for* pk*, i.e. when* $0 \leq \pi^{-1}_{\mathtt{matrixSeed}(\mathrm{pk})}(i) < c$*. It performs this check before the re-encryption check.*

This enables the simulator to answer decryption queries consistently. Note that it already answers $\perp$ if m has actually been used in a challenge ciphertext, since either the ciphertext is that challenge, or it is invalid. Since

the adversary has no information about the set of possible future challenges except that they are different from existing ones,

$$|w_3 - w_2| \leq q_{\text{dec}} \cdot c/(|\mathcal{M}| - c)$$

Game 4 (Use modified oracle)**.** *This is the same as Game 3, except that* (F, H_b, H_e) *are simulated using* (F, H_b, G_e, G_m)*.*

For each public key pk, if pk is high-bit-injective, then (F, H_b, H_e) are 0-indifferentiable from their simulations using (F, H_b, G_e, G_t), and furthermore that set of oracles has the same distribution as its simulation using (F, H_b, G_e, G_m) as above. Therefore by Corollary 3

$$|w_4 - w_3| \leq 2\sqrt{dq\epsilon_{\text{hi}}}$$

Game 5 (Replace decapsulation oracle)**.** *This is the same as Game 4, except that instead of recovering* m *using Dec*(sk, ct)*, it is recovered as*

$$\text{Extract}(\text{pk}, \text{ct}) := m \oplus (F \oplus G_m)(\text{high}_4(\text{ct}))$$

These two games will diverge if and only if the m recovered for some ciphertext is different from that recovered by Dec, and at least one of them encrypts to ct, and ct is not a challenge ciphertext. This can happen in one of two ways: either the adversary submits a ciphertext such that decryption fails, or one such that

$$G_t(\text{pk}, m) \neq (F \oplus G_m)(\text{high}_4(M \cdot H_b(\text{pk}, m)))$$

For any challenge public key and any message, the former happens with probability at most $\delta_{\max}$, and the latter happens with probability ϵ_t.

Let $\mathcal{B}$ be an algorithm which runs $\mathcal{A}$ in the simulator, but if at any point $\mathcal{A}$ submits a decapsulation query $\text{Decaps}(\text{pk}, \text{ct})$ such that $\text{Dec}(\text{sk}, \text{ct}) \neq \text{Extract}(\text{pk}, \text{ct})$, and at least one of these m satisfies $\text{Encaps}(\text{pk}, m) = \text{ct}$, then it exits and returns m. That is, $\mathcal{B}$ is an algorithm which performs an unstructured search for decryption failures. Applying Corollary 1 to $\mathcal{B}$, we

have:[1]

$$\begin{aligned} |w_5 - w_4| &\leq \Pr[\mathcal{B} \text{ finds a decryption failure } m] \\ &\leq 4dq(\delta_{\max} + \epsilon_t) \end{aligned}$$

As of Game 5, the simulator doesn't need the private key.

Game 6 (Puncture at challenge messages)**.** *Game 6 is the same as Game 5, but the simulator punctures all the adversary's queries to hash functions at* (pk, m) *for* $m \in M_{\mathrm{pk}}$*. If* Find *occurs, then the game is a draw. The simulator still calls those hash functions itself to generate challenge messages and shared secrets. This change is irrelevant to decryption queries due to the "no prescient queries" rule.*

By Lemma 2,

$$|w_6 - w_5| \leq \sqrt{(d+1) \cdot P_{\mathrm{find}}}$$

where P_{find} is the probability that $\mathcal{A}$ is measured to have queried such (pk, m). We will evaluate P_{find} over the next few games. Let $P_{\mathrm{find},i}$ be the value of P_{find} in Game i. By Lemma 1 the adversary's view is independent of the shared secrets unless the game is a draw.

Therefore $w_6 = \frac{1}{2}$. Over the next few games, we will study P_{find}.

Game 7 (Coins at random)**.** *Game 7 is the same as Game 6, but the simulator chooses the coins for challenge messages and shared secrets at random.*

At this point all the challenge plaintexts are all distinct for each key, and the public keys are all high-bit-injective, or else the game is a draw. Therefore the hash arguments used to generate challenge messages and shared secrets are distinct, and the simulated hash $\hat{H}_e^{F,H_b,G_m}$ is indeed uniformly and independently random. Thus the challenge coins are independent of each other, and of all other outputs of H_b and $\hat{H}_e$; and the same is true for

[1] It should be noted that each oracle query $H_b(\mathrm{pk}, m)$ in $\mathcal{A}$ is simulated as $\hat{H}_e^{F,H_b,G_m}(\mathrm{pk}, m)$ using multiple oracle queries in $\mathcal{B}$. Naïvely this should increase the d and q used here to greater values such as $2d$ and $2q$. But in fact, treating $(F, H_b, \hat{H}_e^{F,H_b,G_m}, G_m)$ as jointly-distributed oracles shows that dq suffices. Also note that we use dq instead of $(d+1)(q+1)$, because the decapsulation query itself counts toward d and q in our setting, but not in Corollary 1.

shared secrets. By Lemma 1, the adversary's view is independent of these punctured values unless Find, and if Find occurs the game is also a draw. Therefore $P_{\text{find},7} = P_{\text{find},6}$.

Game 8 (IND-CPA-U, $b = 1$)**.** *Game 8 is the same as Game 7, but the simulator uses the IND-CPU-U oracle for* ThreeBears$_r$ *with* $b = 1$ *to create public keys and encrypt challenge messages.*

As of Game 7, the simulator is taking the same steps as the IND-CPA-U oracle, so $P_{\text{find},8} = P_{\text{find},7}$.

Game 9 (IND-CPA-U, $b = 0$)**.** *Game 9 is the same as Game 8, but the simulator uses the IND-CPA-U oracle with* $b = 0$ *to create public keys and encrypt challenge messages.*

By definition, the simulator $\mathcal{B}$ in Game 9 and Game 8 is an IND-CPA adversary against ThreeBears$_r$, so that

$$|P_{\text{find},9} - P_{\text{find},8}| = \text{Adv}_{\text{IND-CPA-U}}(\mathcal{B} : \text{ThreeBears}_r)$$

Furthermore, the values sent to the adversary are independent of the chosen messages, so that by Lemma 3

$$P_{\text{find},9} \leq 4cq/\left|\mathcal{M}\right|$$

Combining the previous several games, we have

$$P_{\text{find},6} \leq \text{Adv}_{\text{IND-CPA-U}}(\mathcal{B}) + 8cq/\left|\mathcal{M}\right|$$

6.5 Summing up

Finally, we have shown that

$$\begin{aligned}\left|w_0 - \frac{1}{2}\right| &\leq \sqrt{(d+1)\cdot(\text{Adv}_{\text{IND-CPA-U}}(\mathcal{B} : \text{ThreeBears}_r) + 8cq/\left|\mathcal{M}\right|)} \\ &+ 4dq(\delta_{\max} + \epsilon_t) \\ &+ \frac{c\cdot c_{\text{tot}}}{2\left|\mathcal{M}\right|} + \frac{c\cdot q_{\text{dec}}}{\left|\mathcal{M}\right| - c} + 2\sqrt{dq\epsilon_{\text{hi}}} + \frac{1}{2}k\epsilon_{\text{hi}}\end{aligned}$$

so the adversary's IND-CCA-U advantage is at most twice this value, as claimed.

- $\sqrt{(d+1) \cdot (\mathrm{Adv}_{\mathrm{IND-CPA-U}}(\mathcal{B} : \textsc{ThreeBears}_r)}$ is an IND-CPA attack, amplified by $d+1$ due to the removal of the circular dependency of the coins on m. We conjecture that this term is loose, but we cannot prove it.
- $\sqrt{(d+1) \cdot (\ldots + 8cq/\left|\mathcal{M}\right|)}$ corresponds to a Grover attack to recover the ciphertext seed for some challenge message.
- $4dq(\delta_{\max} + \epsilon_t)$ corresponds to a Grover attack to find either a message that fails to decrypt, or where our extractor fails.
- $\frac{c \cdot c_{\mathrm{tot}}}{|\mathcal{M}|}$ is the probability that the adversary wins immediately due to two challenge messages colliding.
- $\frac{c \cdot q_{\mathrm{dec}}}{|\mathcal{M}|-c}$ does not correspond to a known attack, but prevents the simulator from becoming inconsistent when faced with prescient decryption queries.
- $2\sqrt{dq\epsilon_{\mathrm{hi}}} + \frac{1}{2}k\epsilon_{\mathrm{hi}}$ does not correspond to a known attack, but to the probability that public keys aren't high-bit-injective.

6.6 One-way vs indistinguishability

The reader might notice that we have reduced to IND-CPA security of the underlying PKE, and not OW-Passive security. But it only uses the challenge messages m to test if the adversary is querying them in the hash. This is for a rather trivial reason: the underlying PKE is randomized. If we were to reduce to OW-Passive, the simulator in Game 6 would not be able to test whether a given $H(\mathrm{pk}, m)$ is querying a challenge message. Instead we would need to abort on a random query and output m as a guess. The theorem would follow from Lemma 4, but with an additional looseness factor of q.

To modularize Theorem 2 to other systems, we could use a notion of "OW-qPCA1", i.e. one-wayness where the adversary is given a quantum-accessible oracle $\mathrm{PCA}(m)$ that checks whether m is a challenge message. For a deter-

ministic PKE, this is the same as OW-Passive security, and for a randomized PKE it can instead be proved from IND-CPA as in this work.

7 Conclusions

We have shown that if the I-MLWE is hard, then ThreeBears is secure in the quantum random oracle model. Our proof is not perfectly tight, as it loses a square root and a factor of d in the IND-CCA reduction, but this is comparable to most other post-quantum security proofs. Our CCA reduction considers attacks using k keys and c_{tot} messages, though ultimately in the I-MLWE reduction these produce a $k + c_{\text{tot}}$ factor.

8 Acknowledgements

This paper is based on joint work with Dominique Unruh and Andris Ambainis. Thanks to Daniel Kane and Eike Kiltz for helpful discussions. Thanks to Mark Marson for his feedback on a draft of this paper.

References

[AHU18] Andris Ambainis, Mike Hamburg, and Dominique Unruh. Quantum security proofs using semi-classical oracles. Cryptology ePrint Archive, Report 2018/904, 2018. https://eprint.iacr.org/2018/904.

[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, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 41–69. Springer, Heidelberg, December 2011. doi:10.1007/978-3-642-25385-0_3.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer, Heidelberg, August 1999. doi:10.1007/3-540-48405-1_34.

[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, Part I*, volume 10677 of *LNCS*, pages 341–371. Springer, Heidelberg, November 2017. doi:10.1007/978-3-319-70500-2_12.

[JZC+17] Haodong Jiang, Zhenfeng Zhang, Long Chen, Hong Wang, and Zhi Ma. Post-quantum IND-CCA-secure KEM without additional hash. Cryptology ePrint Archive, Report 2017/1096, 2017. https://eprint.iacr.org/2017/1096.

[JZM19a] Haodong Jiang, Zhenfeng Zhang, and Zhi Ma. Key encapsulation mechanism with explicit rejection in the quantum random oracle model. Cryptology ePrint Archive, Report 2019/052, 2019. https://eprint.iacr.org/2019/052.

[JZM19b] Haodong Jiang, Zhenfeng Zhang, and Zhi Ma. Tighter security proofs for generic key encapsulation mechanism in the quantum random oracle model. Cryptology ePrint Archive, Report 2019/134, 2019. https://eprint.iacr.org/2019/134.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, May / June 2010. doi:10.1007/978-3-642-13190-5_1.

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor,

TCC 2004, volume 2951 of *LNCS*, pages 21–39. Springer, Heidelberg, February 2004. doi:10.1007/978-3-540-24638-1_2.

[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, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 520–551. Springer, Heidelberg, April / May 2018. doi:10.1007/978-3-319-78372-7_17.

[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, Part II*, volume 9986 of *LNCS*, pages 192–216. Springer, Heidelberg, October / November 2016. doi:10.1007/978-3-662-53644-5_8.

[XY18] Keita Xagawa and Takashi Yamakawa. (tightly) qcca-secure key-encapsulation mechanism in the quantum random oracle model. Cryptology ePrint Archive, Report 2018/838, 2018. https://eprint.iacr.org/2018/838.

A Proof of Lemma 7

Lemma 7. *Denote by* $\epsilon_{\rm hi}$ *the probability that a* ThreeBears$_0$ *public key fails to be high-bit-injective. Then for recommended* ThreeBears *instances,* $\epsilon_{\rm hi} \leq 2^{-409}$.

Proof. Consider a pk that is not high-bit-injective, and let (m_0, m_1) be messages that produce the collision on values

$$(B_0, B_1) = (M\vec{b}_0 + \vec{e}_0, M\vec{b}_1 + \vec{e}_1)$$

where $(b_0, b_1) = (H_b(m_0), H_b(m_1))$ and let

$$B_\delta := (Mb_0 + e_0) - (Mb_1 + e_1))$$

Let $(y_{0,i}, y_{1,i}, y_{\delta,i})$ be the ith of B_0, B_1 and B_δ respectively. Then

$$y_{\delta,i} = y_{0,i} - y_{1,i} + c \bmod 2^{10}$$

where the carry c is in the range $[-2, 2]$. If $y_{0,i}$ and $y_{1,i}$ have the same upper bits, then $y_{0,i} - y_{1,i} \in [-63, 63]$ so that

$$y_{\delta,i} \in [0, 65] \cup [2^{10} - 65, 2^{10-1}]$$

Therefore if two messages m_0 and m_1 result in the same input to F, then either $\vec{b_0} = \vec{b_1}$, or $b_0 \neq b_1$ but $M(b_0 - b_1)$ lies in a set of at most $(2 \cdot 65 + 1)^{312}$ values. The first case occurs with probability less than $2^{-920} - 2^{-925}$ for all recommended instances. The second case happens with probability less than 2^{-925d}. Summing the probability of these events over all $< 2^{511}$ message pairs (m_0, m_1), we find the overall probability of any of these events to be $< 2^{-409}$. □

B Proof of Lemma 6 security for ThreeBears$_r$

Lemma 6. *Let $\mathcal{A}$ be an IND-CPA adversary against* ThreeBears$_r$*, which instantiates k public keys and at most c challenge ciphertexts per key. Then*

$$\begin{aligned}&\mathrm{Adv}_{\mathrm{IND-CPA-U}}(\mathcal{A} : \textsc{ThreeBears}_r) \\ &\qquad \leq 2k(c+1) \cdot (\mathrm{Adv}_{((\mathrm{d}+1)\times\mathrm{d})-\mathrm{I-MLWE}}(\mathcal{B}) + 2^{-1555})\end{aligned}$$

Proof. This is a standard hybrid argument with $k(c+1)$ options. The simulator $\mathcal{B}$ receives an I-MLWE challenge (M, A, B, C), where

- M is a uniform $d \times d$ matrix.
- A is either $Ms + e$ or a d-long random vector, where $s, e \leftarrow \chi^d \times \chi^d$.
- B is a uniform $1 \times d$ matrix.
- C is either $Bs + e'$ where $e' \leftarrow \chi$, or a random element of the ring R.

For each game $i \leq k$, replace the first $i-1$ public keys with random elements and the ith with (seed, A), and reprogram the matrix expansion hash so that $\texttt{uniform}(\text{seed}) = M$. Generate the rest of the keys honestly.

For each game $(i+1)k + j$ with $j < k$, replace the jth public key with $(\text{seed}, B^\top)$, reprogram the matrix expansion hash such that $\texttt{uniform}(\text{seed}) = M^\top$, and set the i'th ciphertext to that key with $(A, \text{encode}(C, m))$. Replace the earlier ciphertexts with uniformly random strings, and generate the later ones honestly.

Replacing $d \times d \leq 16$ uniform elements of $\{0,1\}^{3120}$ with n uniform elements of $\mathbb{Z}/N$ produces a distribution which differs by less than $n/2^{-1559}$, so the total deviation from uniformity is less than $n/2^{-1555}$.

After $k(c+1)$ such games, all challenge public keys and challenge ciphertexts are uniformly random strings, so the adversary's view is independent of the challenge bit b. Therefore the adversary wins with probability exactly $\frac{1}{2}$. Thus the overall win probability is at most $\frac{1}{2} + k(c+1) \cdot \text{Adv}_{\text{I-MLWE}}(\mathcal{B})$. □

C Proof of Lemma 5

Lemma 5. *If $\mathcal{A}$ is an IND-CPA resp. IND-CCA adversary against* ThreeBears$_0$ *with advantage ϵ, and suppose that it requests at most k public keys. Then there is an IND-CPA resp. IND-CCA adversary $\mathcal{B}$ against* ThreeBears$_0$*, running in about the same time as $\mathcal{A}$, that has IND-CPA-U resp. IND-CCA-U advantage*

$$\epsilon_u \geq \max\left(\epsilon - \binom{k}{2}/s,\quad \epsilon/2 - \binom{k}{3}/s^2,\quad \epsilon/3 - \binom{k}{4}/s^3,\quad \ldots\right)$$

so that

$$\epsilon \leq \min\left(\epsilon_u + \frac{k^2}{2s^1},\quad 2\epsilon_u + \frac{k^3}{3s^2},\quad 3\epsilon_u + \frac{k^4}{4s^3},\quad \ldots\right)$$

Proof. The algorithm $\mathcal{B}$ generates k keys itself and throws away the results, but records how many collisions occur (or efficiently simulates this step).

Suppose that the most common seed occurs n times. This happens with probability at most $\binom{k}{n}/s^{n-1}$.

The simulator chooses a uniformly random number j from 1 to n inclusive, and instantiates and IND-CPA-U resp. IND-CCA-U challenger. It asks this challenger for $m \leq k$ keys, where m is the number of seeds that occurred at least j times. It then generates the $k - m$ other keys with seeds that match the collision counts from the first step. It returns all these keys in a random order. When answering queries for the $j - 1$ keys generated for each seed, it answers as though the challenge bit $b = 0$. For the jth key it forwards to the challenger, and for the $j + 1$st and onward it answers as though $b = 1$.

By the standard hybrid argument, for all integers m, $\mathcal{B}$'s advantage is $\epsilon_u \geq \epsilon/m - \delta$, where $\delta \leq \binom{k}{m+1}/s^m$ is the probability that it instantiates more than m oracles. □

D Proof of Lemma 8

Lemma 8. *For a certain uniformly random oracle family F, there is a pair of efficient hash functions $G_e(\mathrm{pk}, m), G_t(\mathrm{pk}, m)$ such that if* pk *is high-bit-injective, then:*

- *(F, H_b, H_e) is 0-indifferentiable from (F, H_b, G_e, G_t).*
- *G_t is a uniformly random oracle which returns a 256-bit value.*
- *$G_t(\mathrm{pk}, m)$ can be calculated from only the ciphertext component $B = Mb + e$, except with probability at most $\epsilon_t < 2^{-319}$ independently per message.*

Proof. Let $F(\mathrm{pk}, h)$ be a random oracle which takes a $(4 \cdot 312 \cdot d)$-bit input, and returns a uniformly random $(6 \cdot 312 \cdot d)$-to-256-bit affine function over $\mathbb{F}_2$. The idea is to set

$$\hat{G}^H_{t,0}(\mathrm{pk}, m) := F(\mathrm{high}_4(B))(\mathrm{low}_6(B)) \text{ where } B = MH_b(m) + H_e(m)$$

For most messages m, $\hat{G}_t^H(\mathrm{pk}, m) = \hat{G}_{t,0}^H(\mathrm{pk}, m)$. To calculate the actual $\hat{G}_t^H(\mathrm{pk}, m)$, choose a random vector $\vec{v} \in \mathrm{codomain}(F_r)$ where F_r is a certain restriction of F. Then

$$\hat{G}_t^H(\mathrm{pk}, m) \begin{cases} \hat{G}_{t,0}^H(\mathrm{pk}, m) & \text{if } \vec{v} \in \mathrm{codomain}(F_r) \\ \vec{v} & \text{otherwise} \end{cases}$$

For the vast majority of (pk, m), the value $\hat{G}_t^H(\mathrm{pk}, m) = \hat{G}_{t,0}^H(\mathrm{pk}, m)$ can be computed from B. If pk is high-bit-injective, then F is an independent random affine function for each message and $\hat{G}_t^H(\mathrm{pk}, \cdot)$ is a uniformly random oracle.

We now show how to construct G_e and how to emulate H_e using G_e and G_t. Recall that each digit of the noise is chosen as

$$\chi_v := \begin{cases} -1 & \text{with probability } v/2 \\ 0 & \text{with probability } 1 - v \\ +1 & \text{with probability } v/2 \end{cases}$$

or, in the case of BabyBear, as $\chi_{1/2} + \chi_{v-1/2}$.

G_e begins by running H_b to produce b, calculating Mb, and choosing certain parts of each digit e_i of e:

- For BabyBear G_e chooses the coins for $\chi_{v-1/2}$ and sets $v \leftarrow 1/2$.
- G_e sets each $e_i = 0$ with probability $1 - 2v$.
- For the digits that aren't forced to 0, G_e chooses a sign $s_i \in \{\pm 1\}$; e_i will be either 0 or s_i.
- Let P_{ij} be the condition that high 4-bits of the ith coefficient of $Mb+e$ still depends on e_i. With negligible probability ($< 2^{-1559}$) the graph is cyclic, in which case G_e chooses the rest of e immediately. Otherwise in topological order, G_e determines whether P_{ii} holds (it does with probability $1 - 1/64$) and if so it chooses e_i at random from $\{0, s_i\}$.

Call a variable e_i *free* if it is still undetermined by the above procedure, because P_{ii} doesn't hold. Assuming the above graph is acyclic, the probability

that f variables are free is then at least

$$\binom{312 \cdot d}{f} \cdot p^f \cdot (1-p)^{312 \cdot d - f}$$

where $p = v - 1/64$. Each bit of $\mathrm{low}_6(Mb + e)$ is either fixed, or is equal to one of the free variables or its negation. Since an f-to-r bit affine equation is soluble with probability at least $1 - 2^{r-f}$, there is at least one preimage of $G_t(\mathrm{pk}, m)$ with probability at least

$$1 - 2^{-1559} - \sum_{f=256}^{n} \binom{312 \cdot d}{f} \cdot p^f \cdot (1-p)^{312 \cdot d - f} > 1 - 2^{-319}$$

where $(d, p) = (2, 1 - \frac{1}{64}),\ (3, \frac{13}{16} - \frac{1}{64}),\ (4, \frac{9}{16} - \frac{1}{64})$ for BabyBear, MamaBear and PapaBear respectively.

Let E_f be the subspace $\mathbb{F}_2^f$ of all the remaining choices of e, with a 0-component meaning $e_i = 0$ and 1 component meaning $e_i = s_i$. Let e_f denote the actual choices made for a given e. Let F_r be the map $F(\mathrm{pk}, m)$ restricted to E_f. G_e outputs a random element $k \in \ker F_r \subset E_f$. Finally, it samples $\vec{w} \leftarrow G_t(\mathrm{pk}, m)$; if $\vec{w} \notin \mathrm{range}(F_r)$, then G_e outputs $\vec{v} \leftarrow \vec{w}$; otherwise, it outputs $\vec{v}$ uniformly at random in $\mathrm{range}(F_r)$.

The map $\hat{G}_e^H(\mathrm{pk}, m)$ performs the above steps, but at each step it chooses the restriction on e_i which is consistent with $e := H_e(\mathrm{pk}, m)$. For the final step, if $\vec{v} \in \mathrm{range}(F_r)$ the algorithm chooses $k = e - e_0$, where e_0 is the lexicographically least preimage of $\hat{G}_e^H(\mathrm{pk}, m)$ under F_r.

In the opposite direction, the map $\hat{H}_e^G(\mathrm{pk}, m)$ calculates e by running G_e, which determines the high bits of $Mb+e$; it then calculates $F(\mathrm{high}_4)(Mb+e)$ and F_r. If $G_t(\mathrm{pk}, m) \in \mathrm{range}(F_r)$, it outputs $\vec{v} \xleftarrow{R} \mathrm{range}(F_r)$ and finds e_0 as the lexicographically least solution to

$$F(\mathrm{high}_4)(e_0) = G_t(\mathrm{pk}, m)$$

which is consistent with the choices made by G_e. It chooses $e_f = e_0 + k$. If, on the other hand, $G_t(\mathrm{pk}, m) \notin \mathrm{range}(F_r)$, then $\hat{H}_e^G(\mathrm{pk}, m)$ outputs $\vec{v} = G_t(\mathrm{pk}, m)$, and e_f uniformly at random. This completes the indifferentiability construction. □