Key Encapsulation from Noisy Key Agreement in the Quantum Random Oracle Model

Alan Szepieniec[1], Reza Reyhanitabar[2], and Bart Preneel[1]

[1] imec-COSIC KU Leuven, Belgium
alan.szepieniec@esat.kuleuven.be, bart.preneel@esat.kuleuven.be
[2] Elektrobit Automotive GmbH, Germany
reza.reyhanitabar@elektrobit.com

Abstract. A multitude of post-quantum key encapsulation mechanisms (KEMs) and public key encryption (PKE) schemes *implicitly* rely on a protocol by which Alice and Bob exchange public messages and converge on secret values that are identical *up to some small noise*. By our count, 24 out of 49 KEM or PKE submissions to the NIST Post-Quantum Cryptography Standardization project follow this strategy. Yet the notion of a *noisy key agreement* (NKA) protocol lacks a formal definition as a primitive in its own right. We provide such a formalization by defining the syntax and security for an NKA protocol. This formalization brings out four generic problems, called A and B State Recovery, Noisy Key Search, and Noisy Key Distinguishing (NKD), whose solutions must be hard in the quantum computing model. Informally speaking, these can be viewed as noisy, quantum-resistant counterparts of the problems arising from the classical Diffie-Hellman type protocols. We show that many existing proposals contain an NKA component that fits our formalization and we reveal the induced concrete hardness assumptions. The question arises whether considering NKA as an independent primitive can help provide modular designs with improved efficiency and/or proofs. As the second contribution of this paper, we answer this question positively by presenting a generic transform from a secure NKA protocol to an IND-CCA secure KEM in the quantum random oracle model, with a security bound related to the insecurity of the NKD problem. This transformation is essentially the same as that of the NIST candidate Ramstake. While establishing the security of Ramstake was our initial objective, the collection of tools that came about as a result of this journey is of independent interest.

Keywords: Post-quantum, key encapsulation, public key encryption, quantum random oracle model, noisy key agreement.

1 Introduction

Post-Quantum Cryptography. Most of the standard public key cryptosystems in use, including Diffie-Hellman and derivatives thereof, RSA, DSA, ECDSA,

and ElGamal cryptosystems, rely on the computational hardness of number theoretic problems. For these problems, in particular factoring and discrete log (DLOG) problems, quantum computers offer exponential speedups compared to classical computers. Shor's factoring and discrete logarithm algorithms [57] render these cryptosystems insecure in the quantum computing era. The anticipation of this threat is what drives the development and deployment of *post-quantum cryptography*—cryptographic algorithms that despite running on classical computers promise to resist quantum attacks—well before large-scale quantum computers arrive.

In contrast to the aforementioned public key schemes, symmetric key algorithms such as AES and its various modes of operations, as well as hash functions such as SHA2 and SHA3 remain relatively unaffected by quantum computers. The best known quantum attack on these primitives is Grover's generic search algorithm [33] and it offers only a square root speed-up, meaning that the same security level is attained against quantum computers by merely doubling the key or output length.

In response to this highly selective quantum threat, the American National Institute for Standards and Technology (NIST) has issued a call for proposals covering some public key functionalities: key encapsulation mechanisms (KEMs), public key encryption (PKE) schemes, and signature schemes. The intention is to issue a post-quantum cryptography standard as a result of this process [49]. Out of 69 complete and proper submissions, 22 proposals achieve signature scheme functionality and 49 achieve KEM or PKE or both (with some overlap) [50].

Key Exchange (KE). KE protocols enable two parties who communicate over an adversarially-controlled channel to obtain a secret session key. Starting with the seminal work of Diffie and Hellman [28], there is now a rich body of work on this topic in the literature containing several security models and design paradigms [12,20,41,42,24]. By convention, we consider Key Agreement (KA) protocols as a subset of KE protocols in which both parties influence the generation of the resulting session key; for instance, Diffie-Hellman (DH) type protocols are classic examples of KA.

Key Encapsulation Mechanism. Cramer and Shoup [22,23] provided, among other contributions, a formal treatment of hybrid Public Key Encryption (PKE) secure against adaptive chosen ciphertext attacks (CCA) [56]. The approach, known as the KEM/DEM (Key Encapsulation Mechanism/Data Encapsulation Mechanism) framework, rigorously captures the folklore method for building a hybrid encryption scheme, namely by using public key cryptography to *encapsulate* a symmetric *session key*, followed by symmetric-key encryption.

While the original and main application of KEM has been in hybrid PKE, it has turned out that pure KEM can be a useful cryptographic tool in its own right in other applications; for example, to build schemes for identification [9] and authenticated key exchange [19,31,66].

Design Strategies. We identify three binary design choices that partition the design space of KEMs and PKEs. They are *noisy versus noise-free*, *convergence versus inversion*, and *reconciliation versus transmission*. The last choice only makes sense in the case of noisy convergence.

Noisy versus noise-free considers the nature of the underlying mathematical hard problems. Multivariate quadratic (MQ) equations and supersingular isogenies (SI) achieve computational hardness without adding random noise, whereas lattice- and code-based problems are computationally difficult precisely because they rely on the addition of noise. The newest member of the latter class is the family of problems based on sparse integers and arithmetic modulo (pseudo-) Mersenne primes [1,50].

Convergence versus inversion looks at the strategy to achieve the targeted KEM or PKE functionality. The earlier MQ, code- and lattice-based cryptosystems relied on trapdoor inversion [45,46,51,34], in which the public operation amounts to evaluating a trapdoor function and the secret operation amounts to inverting it. In contrast, newer proposals implicitly rely on a noisy key agreement protocol in which two parties obtain *roughly* the same key which is hard for the passive eavesdropper to approximate [29,7,16,26]. The exception to this rule is the supersingular isogeny Diffie-Hellman (SIDH) cryptosystem [38], and its brother CSIDH [21], both of which converges on identical keys and hence might be termed an *exact key agreement* (EKA) protocol but nevertheless amounts to a special case of NKA. To date, SIDH and CSIDH are the only post-quantum cryptosystem capable of achieving *static key agreement* (SKA) functionality, whereby any pair of participants who know each other's public key can derive the same shared symmetric key *without interaction*, opening up the possibility for bypassing the exchange of public key messages and instead communicating over the symmetric channel immediately.

Reconciliation versus transmission deals with the details of obtaining identical keys after similar keys were obtained through a noisy convergence strategy. Reconciliation entails sending helper data to enable the receiver to correct the errors or otherwise extract an identical template from the noisy views of the shared key. There are many subtle variants, all of which rely on the specific mechanics of the underlying mathematics [29,54,16,62]. In contrast, transmission[3] uses the shared noisy key to mask a new message entirely; this new message must then contain enough redundancy to be decodable after being masked and unmasked with two approximately equal one-time pads. Transmission is arguably less prone to error, but does come with a bandwidth penalty [6,40].

Our Contribution. This paper presents two main contributions. The first is a formal syntax and security definition to capture the notion of a noisy key agreement (NKA) protocol as a new useful primitive. The second is a generic transformation to turn an NKA protocol into an IND-CCA secure KEM in the quantum random oracle model. Based on the previous categorization of design

[3] Also called the *encryption-based approach* in NewHopeSimple [6], and an *asymmetric key consensus* in the context of OKCN/AKCN [40].

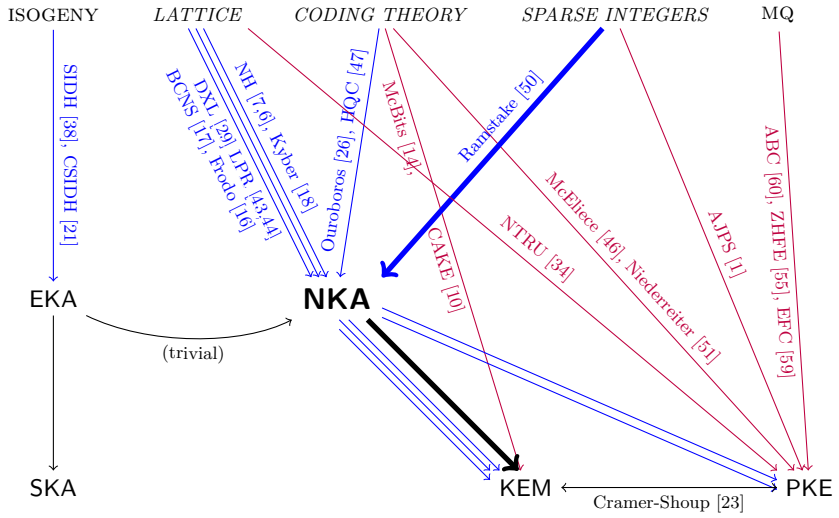

Fig. 1: Map of post-quantum KEM and PKE. The bold objects indicate the contributions of this paper. Italics denotes noisy mathematics; blue arrows denote convergence, red ones denote inversion, and black ones represent generic transforms.

strategies, our transformation applies to noisy convergence based protocols, and uses the transmission strategy.

The syntax of NKA protocols captures the intuition where, after an initialization phase that generates public parameters, Alice and Bob generate a state and contribution pair. They then exchange their protocol contributions and use their own state and the other party's contribution to converge on approximately the same value. An explicit treatment of protocol failure events resulting from excessive noise, which may cause decryption or decapsulation errors, is built in to our formalism.

This syntax naturally lends to four attack vectors, which we formulate as generic problems called *A State Recovery (ASR)*, *B State Recovery (BSR)*, *Noisy Key Search (NKS)*, and *Noisy Key Distinguishing (NKD)*, mirroring the DLOG, computational Diffie-Hellman (CDH) and decisional Diffie-Hellman (DDH) Problems in Diffie-Hellman protocols. While the classical DLOG, CDH and DDH problems are efficiently solvable by quantum algorithms, these new generic problems arising from formalization of *noisy* key agreement must remain hard in the quantum computing model. Hence, instantiations of NKA must rely on concrete hardness assumptions that guarantee infeasibility of these generic problems even in the face of quantum solvers. Many existing proposals contain an NKA component that fits our formalization; we identify the induced concrete hardness assumptions.

Security of an NKA protocol is defined with respect to the NKD problem. Specifically, an NKA protocol is secure if and only if its NKD problem is hard on average. We justify this definition in several ways.

– The hardness of NKD implies the hardness of NKS, ASR and BSR; therefore the NKD Assumption is the strongest assumption.
– It is analogous to regular Diffie-Hellman, where the protocol is secure if and only if the DDH problem is hard (assuming authenticated links).
– We consider an example from the NIST PQC project that fits the NKA framework and where ASR and BSR are hard, but where NKD is easy, which led to the submission's prompt cryptanalysis.
– We consider in the appendix an alternate definition of security based on a suitable adaptation of the well-known Canetti-Krawczyk session-key security (SK-security) notion [20]. We find that this security notion is equivalent to the average-case hardness of the NKD problem.

These results indicate that the average-case hardness of the NKD problem is essential in the context of secure NKA-based KEMs and PKEs.

As our second and main contribution, we provide a generic NKA-to-KEM transformation for *noisy*, *convergence*-based protocols, applying the *transmission* strategy, and featuring an IND-CCA security proof in the quantum random oracle model. The main feature in this context is its *genericity*: it applies regardless of the mathematics of the underlying NKA protocol and as such enables a modular design workflow. We note that the Ramstake submission [50] uses essentially the same transformation but was presented without proof; this paper therefore proves the security of Ramstake, assuming the appropriate NKD problem is hard on average. In comparison to other IND-CCA transforms in the literature, the most obvious difference is that the starting point of our transform is an NKA protocol, whereas other IND-CCA transforms start from an IND-CPA secure PKE or KEM. We include the key-confirmation hash of Targhi-Unruh in the ciphertext [61] and follow the *derandomization and re-encryption* approach so named by Hofheinz *et al.* [35]. We note that a recent result by Jiang *et al.* [39] suggests that this additional hash might not be necessary, but we leave open for the time being the question whether dropping it affects the security of our particular construction. In contrast to these related results [61,35,39], our session key is computed from *bipartite contribution*, *i.e.*, as a function of both the public key and the encapsulator's randomness; this property prevents Bob from establishing the same symmetric key for separate channels, one with Alice and one with Charlie.

An outstanding feature of our proof is the tighter security bound: the insecurity of the underlying primitive (NKD of the NKA protocol in our case; IND-CPA security of the PKE or KEM elsewhere) undergoes a square-root degradation, similar to the result by Jiang *et al.* and in stark contrast to the quartic root degradation of Targhi-Unruh and Hofheinz *et al.*. This improved bound is the result of treating the extendable output function that is used for derandomization as a random oracle; this enables an argument about the queries that are made to it. While our bound does feature fourth-roots, they apply only to the hash function insecurity.

Central to our security proof is a new technique for lifting classically-valid random oracle model security proofs to the quantum random oracle model. We

introduce, define, and use, the *aggregate quantum query amplitude*, which behaves similar to the expected number of times a particular query was made by an adversary throughout the entire computation. We use this notion as a starting point to derive lemmata that enable refined argumentation about adversarial query behavior, as well as to derive a multi-target generalization of Unruh's One-Way to Hiding Lemma [64]. These lemmata are used in the security proof to capture the intuition that a quantum adversary does not know the random oracle's output on inputs that were not queried. We believe this notion and our proof technique to be of independent interest as a useful tool in security analysis of other PQC schemes.

Ramstake and the NIST PQC Project. While our starting point was the establishment of a security proof for Ramstake, this journey has led to many independently useful tools for the analysis and provable security of post-quantum cryptosystems. Nevertheless, we stress that despite the detour we were successful in this endeavor. The main contribution of this paper remains the establishment of a security proof reducing the IND-CCA security of Ramstake to solving the appropriate version of the NKD problem — called Low Hamming Diffie-Hellman Decision (LHDHD) Problem in the context of Ramstake [50].

The ongoing NIST PQC project, as a design-focused project with a somewhat fixed timeframe, has boosted research on PQC and has attracted 69 proposals, which are the subject of intense scrutiny. Nevertheless —or perhaps accordingly— it is compelling and timely to revisit the foundations of security notions and of design paradigms for next-generation PQC schemes in order to stay ahead of emerging threats and to prevent past failures from being transmuted in future. This paper aims to be a step forward in this direction.

Organization of the Paper. Section 2 provides notations, conventions and definitions used throughout the paper. In Sect. 3 we present our *noisy key agreement* formalism, including syntax, abstract hard problems, and security definition. Section 4 presents our NKA-to-KEM transformation, and we follow up in Sect. 5 with a discussion on proof techniques (including the aggregate quantum query amplitude) before presenting the security proof.

2 Preliminaries

Notation and conventions. We use $a \leftarrow b$ to denote the assignment of the value b to the variable a, and $a \xleftarrow{\$} A$ to denote the assignment of a uniformly random element from the set A. Algorithms are denoted in sans-serif font and the event that an algorithm A, on input x, outputs y is written as $\mathsf{A}(x) \Rightarrow y$ and $\mathsf{A}(x) \not\Rightarrow y$ when it does not output y. A long double right arrow ($\Longrightarrow$) denotes logical implication, and $\triangleq$ denotes equality by definition. Superscript, *e.g.*, A^{O} denotes an algorithm A having oracle access to O, meaning that A can query O and receive responses in a black box manner but he cannot study the oracle's code or composition.

A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}_{>0}$ is *negligible* if for all polynomials $p(x) \in \mathbb{R}[x]$ there is an $N \in \mathbb{N}$ such that for all $x > N$, $\mathsf{negl}(x)$ drops faster than the reciprocal of $|p(x)|$. Formally, we need only consider the dominant monomial of $p(x)$:

$$\forall c > 1 \,.\, \exists N \in \mathbb{N} \,.\, \forall \lambda > N \,.\, \mathsf{negl}(\lambda) \leq \frac{1}{\lambda^c} \ .$$

Quantum Computation. The state of a quantum system of k qubits is given by a unit-length vector in *ket* notation, *e.g.* $|\Psi\rangle \in \mathcal{H}$, where $\mathcal{H} \subset \mathbb{C}^{2^k}$; where $\langle\Psi|$ is its complex conjugate transpose, and $\langle\Psi|\Phi\rangle$ is the standard inner product. The composition of two quantum systems is described by the tensor product $|\Psi\rangle \otimes |\Phi\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2$, which is the vector of all multilinear products. However, sometimes quantum systems of more than one qubit cannot be factored into the tensor product of independent systems; in this case the two systems are *entangled*. Except for measurements, all quantum computations are unitary transforms on the state space. Measurement of a system $|\Psi\rangle$ is defined with respect to a set of orthonormal basis vectors $|b_0\rangle, |b_1\rangle, \ldots, |b_{2^k-1}\rangle$ and affects the system by collapsing it to $|b_i\rangle$ with probability $\langle b_i|\Psi\rangle\langle\Psi|b_i\rangle$. Any bitstring $s \in \{0,1\}^k$ has an associated basis vector $|s\rangle = |b_i\rangle$ for some i. Whenever a state is a non-trivial sum of basis vectors, *i.e.*, with weights different from 0, -1 and 1, it represents a *superposition* of values. Except for measurement, all quantum operations are reversible. Moreover, it is possible to transform any quantum circuit into an equivalent circuit where all the measurement operators are located at the end.

An equivalent characterization of quantum computation is in terms of a system's density operator or density matrix $\rho \in \mathbb{C}^{2^k \times 2^k}$, as opposed to its state vector $|\Psi\rangle \in \mathcal{H} \subset \mathbb{C}^{2^k}$. The density operator associated with a pure state $|\Psi\rangle$ is $\rho = |\Psi\rangle\langle\Psi|$. When the density operator has a higher rank it represents a probability ensemble: the density matrix $\rho = \sum p_i |\psi_i\rangle\langle\psi_i|$ represents a system that has a probability p_i of having state $|\psi_i\rangle$. The density operator is especially useful for its characterization of parts of a complex quantum system because this operator, together with the partial trace operator, leads to the correct determination of observable statistics. The *reduced density operator* ρ_A of a subsystem A of a composite system $A + B$ with density matrix $\rho_{A,B}$ is obtained by "tracing out" the Hilbert space $\mathcal{H}_B$ associated with B, *i.e.*, by applying the *partial trace* operator $\rho_A = \mathsf{Tr}_B \rho_{A,B}$ which is defined by $\forall |a\rangle \in \mathcal{H}_A, |b\rangle \in \mathcal{H}_B \,.\, \mathsf{Tr}_B(|a\rangle \otimes |b\rangle\langle a| \otimes \langle b|) = |a\rangle\langle a|\langle b|b\rangle$. For more details on quantum computation and quantum information we refer the reader to a comprehensive treatment of the subject by Nielsen and Chuang [52].

We use capital letters without ket notation to denote quantum registers, *i.e.*, the sets of qubits assigned to a variable. We use lowercase letters in ket notation to denote computational basis vectors with unspecified index, and Greek letters in ket notation to denote non-trivial superpositions of computational basis states.

Quantum Random Oracle Model. Our security proof relies on the modeling of hash functions as *random oracles* [30,13], which are uniformly random functions $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$ with a fixed output length, typically equal to the

security parameter. If necessary, the random oracle's output space can be lifted to any finite set X. We use subscripts to differentiate the random oracles associated with different output spaces. The adversary has no access to the function's full description or source code. Security proofs of this type are said to hold in the *random oracle model* (ROM).

Boneh *et al.* show that the random oracle model is not a suitable model when attacks on quantum computers are to be considered [15]. Instead, adversaries have access to a black box that operates on a query-response register pair (Q, R) by sending $|q, r\rangle \mapsto |q, r \oplus \mathsf{H}(q)\rangle$. In this model, quantum adversaries are capable of querying the random oracle on superpositions of bit strings and should receive a superposition answer back. Many classically-valid random-oracle constructions fail to account for this capability and rely in their security proofs on notions or behaviors which become ill-defined when quantum access is considered, such as the list of queries or lazy sampling. As a result, the security proof is valid in the classical random oracle model but invalid in the *quantum random oracle model* (QROM). Many subsequent works elaborate on the notion either by lifting constructions or proofs to the QROM [58,63,65,61], or by showing that such a lift is impossible [25,8].

Derandomization. Our construction relies on derandomization. While pseudorandom generators are usually sufficient for this task, in our case the adversary has quantum oracle access to the function. We therefore opt for an extendable-output function (XOF) [48], which we model as a random oracle in the security proof.

In derandomization, probabilistic polynomial-time algorithms are made deterministic. In particular, let A be a probabilistic polynomial-time algorithm and $s \in \{0,1\}^\lambda$ a seed. We write $\mathsf{A}(x)$ to denote that A is run on input $x \in \{0,1\}^*$, and $\mathsf{A}(x; r)$ to make the contents of its random tape $r \in \{0,1\}^R$ explicit. Then A is derandomized by invoking $\mathsf{A}(x; \mathsf{H}_3(s, R))$ for some s. In fact, in our construction we make abstraction of the output length R and instead use denote by H_3 the function that takes a short input and outputs "enough" random bits.

Key Encapsulation Mechanism. A Key Encapsulation Mechanism (KEM) $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ is a triple of probabilistic polynomial-time algorithms, where

- KeyGen takes a security parameter λ (in unary representation) and outputs two objects: a secret key sk and a public key pk;
- Encaps takes a public key pk and outputs two objects: a symmetric key k from a symmetric key space $\mathsf{SKSpace}$ and a ciphertext c;
- Decaps takes a secret key sk and a ciphertext c and outputs a session key k from the symmetric key space $\mathsf{SKSpace}$, or returns $\perp$ if a failure has occurred.

A KEM's *failure probability* ϵ is defined as

$$\epsilon = \Pr\left[k_e \neq k_d \;\middle|\; \begin{array}{l} sk, pk \leftarrow \mathsf{KeyGen}(1^\lambda) \\ k_e, c \leftarrow \mathsf{Enc}(pk) \\ k_d \leftarrow \mathsf{Dec}(sk, c) \end{array} \right] , \tag{1}$$

and should be small or else the scheme is not usable.

Security of KEMs is defined using the following IND-CCA[4] game, defined with respect to an adversary $\mathsf{A}^{\mathsf{D}(\cdot)}$ who who has black box access to a decapsulation oracle. The IND-CPA game relaxes this notion by disallowing decapsulation queries, but is otherwise identical.

Game 2: IND-CCA

1. $sk, pk \leftarrow \mathsf{KeyGen}(1^\kappa)$
2. $b \xleftarrow{\$} \{0,1\}$
3. $k_0 \xleftarrow{\$} \mathsf{SKSpace}$
4. $c, k_1 \leftarrow \mathsf{Encaps}(pk)$
5. $\mathcal{S} \leftarrow \varnothing$
6. **define** $\mathsf{D}(q)$ **as:**
7. $\quad \mathcal{S} \leftarrow \mathcal{S} \cup \{q\}$
8. $\quad$ **return** $\mathsf{Decaps}(sk, q)$
9. $b' \leftarrow \mathsf{A}^{\mathsf{D}(\cdot)}(pk, k_b, c)$
10. **return** $[\![b = b' \wedge c \notin \mathcal{S}]\!]$

Game 3: IND-CPA

1. $sk, pk \leftarrow \mathsf{KeyGen}(1^\kappa)$
2. $b \xleftarrow{\$} \{0,1\}$
3. $k_0 \xleftarrow{\$} \mathsf{SKSpace}$
4. $c, k_1 \leftarrow \mathsf{Encaps}(pk)$
5. $b' \leftarrow \mathsf{A}(pk, k_b, c)$
6. **return** $[\![b = b']\!]$

The Iverson brackets $[\![\cdot]\!]$ evaluate to 1 if the logical expression is true and to 0 otherwise. A KEM is *secure* if for all polynomial-time quantum adversaries $\mathsf{A}^{\mathsf{D}(\cdot)}$ with *classical* black box query access to a decapsulation oracle D, their advantage $\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathcal{E}}(\mathsf{A}^{\mathsf{D}(\cdot)})$ is negligible:

$$\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathcal{E}}(\mathsf{A}^{\mathsf{D}(\cdot)}) \triangleq \left| \Pr\left[\mathsf{Game}^{\mathsf{A}^{\mathsf{D}(\cdot)}}_{\mathsf{IND\text{-}CCA}}(1^\lambda) \Rightarrow 1\right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda) \ . \tag{2}$$

Most proposals for post-quantum KEMs claim only to satisfy the strictly weaker indistinguishability under chosen plaintext attack (IND-CPA) security notion and emphasize targeting the exchange of *ephemeral keys* only, being a scenario in which chosen ciphertext attacks are unrealistic. Nevertheless, there are several notable exceptions that do meet the stronger IND-CCA requirement [10,18,4]. Moreover, there are generic conversions from IND-CPA secure KEMs and PKEs to IND-CCA secure ones in the classical and quantum random oracle models [32,27,61,36,35].

Error-Correcting Codes. A linear $[n, k, d]$-code $\mathcal{C}$ is a subspace $\mathbb{F}_q^n$ of dimension k. We consider here only bitstrings in which case the *symbol field* $\mathbb{F}_q = \mathbb{F}_2$ and codewords are elements of $\mathbb{F}_2^n \cong \{0,1\}^n$ but encode elements of $\mathbb{F}_2^k \cong \{0,1\}^k$ with $k < n$. The *minimum distance* d of a code is the Hamming weight of its smallest nonzero codeword: $d = \mathsf{min}_{\mathbf{c} \in \mathcal{C} \setminus \{\mathbf{0}\}} \mathrm{HW}(\mathbf{c})$. The code is capable of finding the nearest codeword $\mathbf{c} \in \mathcal{C}$ to a noisy word $\mathbf{c}'$ as long as the Hamming weight of the distance is at most d: $\mathrm{HW}(\mathbf{c}' - \mathbf{c}) \leq d/2$. This process is called *error correction*. This paper abstractly assumes the availability of two functionalities:

[4] The pseudocode of Game 2 follows the IND-CCA-OP notion of Bellare, Hofheinz and Kiltz [11], who prove equivalence between this and five other common IND-CCA notions for KEMs.

– $\mathcal{C}.\mathsf{encode} : \mathbb{F}_2^k \to \mathbb{F}_2^n$, which generates codewords from messages;
– $\mathcal{C}.\mathsf{decode} : \mathbb{F}_2^n \to \mathbb{F}_2^k$, which corrects the errors and returns the associated message, or returns $\perp$ if there are more than some threshold t of errors.

3 Noisy Key Agreement

The common theme in all constructions relying on what we call noisy key agreement is the distinction between "small" and "large" elements in compatible spaces. Before the protocol starts, Alice and Bob agree publicly on a random large element G. When the protocol starts, both generate small secrets a, b and c, d respectively. They then exchange messages $aG + b$ and $cG + d$, and obtain views $acG + ad$ and $acG + cb$ of a noisy shared secret which differ only by a still-small term $ad - cb$.

To the best of our knowledge, the first use of the term "Noisy Diffie-Hellman" traces back to a pair of presentations given by Gaborit in 2010 [2,3], although the underlying strategy was already folklore knowledge by that point[5]. We prefer to reserve the term Diffie-Hellman for noise-free key agreement protocols involving square-and-multiply or double-and-add procedures to compute commutative actions on group elements.

The purpose of this section is to abstract out the mathematics and find a syntax that contains all instances of this principle. We call the resulting formalism *noisy key agreement* (NKA) and it targets a number of desirable properties: (i) NKA should contain standard Diffie-Hellman-based key agreement protocols for noise level zero. (ii) NKA should come with a usable security definition. (iii) NKA should be identifiable inside the constructions that are supposedly based on it.

3.1 Syntax

We formalize the above intuition as follows. Before the protocol starts, Alice and Bob must agree on a set of *instance parameters iparams*, which is the output of the *initialization function* Init when run on the security parameter λ (provided in unary notation). Alice's and Bob's tasks during the protocol are divided into two algorithms each. In the *contribute algorithms* AContr and BContr, they each generate a *state*, *A_state* and *B_state*, in addition to *contributions* *A_contr* and *B_contr*. The contributions are sent to the other party, whereas the states are kept secret. In the *converge algorithms* AConv and BConv, Alice and Bob use their own proper state and the other party's contribution to obtain a view of the shared noisy key: $S_A \leftarrow \mathsf{AConv}(A\_state, B\_contr)$ and $S_B \leftarrow \mathsf{BConv}(B\_state, A\_contr)$. Without loss of generality, we assume that S_A and S_B are bit strings of length ℓ. If all goes well, the two views of the session key are *close*, or specifically, different in at most t bits: $\textsc{hw}(S_A \oplus S_B) \leq t$.

[5] Consider for instance Peikert's invited talk at TCC 2009 [53] or Alekhnovich's FOCS 2003 paper [5].

Definition 1 (noisy key agreement protocol). *A* noisy key agreement *protocol between two parties A and B is a tuple $\Pi = (\mathsf{Init}, \mathsf{AContr}, \mathsf{BContr}, \mathsf{AConv}, \mathsf{BConv})$ of five polynomial-time algorithms where the first three are probabilistic and the last two are deterministic. The algorithms are associated with spaces ParSp, $\mathsf{ContrSp}$, $\mathsf{StateSp}$, $\{0,1\}^\ell$ and have type signatures as follows (omitting the random coins and where λ is the security parameter).*

- $\mathsf{Init} : \{1^\lambda\} \to \mathsf{ParSp}$
- $\mathsf{AContr}, \mathsf{BContr} : \mathsf{ParSp} \to \mathsf{StateSp} \times \mathsf{ContrSp}$
- $\mathsf{AConv}, \mathsf{BConv} : \mathsf{StateSp} \times \mathsf{ContrSp} \to \{0,1\}^\ell$

The algorithms are such that, with respect to a noise level $t \leq \ell/2$ *and* correctness error ϵ,

$$\Pr\left[\mathrm{HW}(S_A \oplus S_B) \leq t \;\middle|\; \begin{array}{l} \mathit{iparams} \leftarrow \mathsf{Init}(1^\lambda) \\ A\_\mathit{state}, A\_\mathit{contr} \leftarrow \mathsf{AContr}(\mathit{iparams}) \\ B\_\mathit{state}, B\_\mathit{contr} \leftarrow \mathsf{BContr}(\mathit{iparams}) \\ S_A \leftarrow \mathsf{AConv}(A\_\mathit{state}, B\_\mathit{contr}) \\ S_B \leftarrow \mathsf{BConv}(B\_\mathit{state}, A\_\mathit{contr}) \end{array}\right] \geq 1 - \epsilon \ , \quad (3)$$

where $\mathrm{HW}\colon \{0,1\}^* \to \mathbb{N}$ *is the Hamming weight function.*

3.2 Generic Problems

The NKA syntax defines three attackable secrets whose recovery is sufficient to undermine the security of the protocol. Also, since the shared secret is what is used in a subsequent module, we note that distinguishing it from random may be a fourth viable attack in many circumstances. We capture these attack strategies in the language of generic problems whose average-case hardness is a necessary condition for security. Any instantiation of NKA therefore defines concrete instantiations of of these hard problems, which then induce concrete average-case hardness assumptions which are necessary for that protocol's security.

The first pair of problems is to recover Alice's secret state from their protocol contribution. If AContr and BContr are identical, then so are these two problems. In the standard Diffie-Hellman key agreement protocol, these problems boil down to the discrete logarithm problem: to obtain a from p, g, and $g^a \bmod p$.

A State Recovery (ASR).
Input: $\mathit{iparams}$, A_contr
Task: find A_state.

B State Recovery (BSR).
Input: $\mathit{iparams}$, B_contr
Task: find B_state.

The next problem captures the task of finding the agreed-upon session key, or a similar enough bit string, from all public information. In the standard Diffie-Hellman key agreement protocol, this problem is essentially the computational Diffie-Hellman problem, *i.e.*, asking to obtain g^{ab} from g, g^a and g^b (all $\bmod p$).

Noisy Key Search (NKS).
Input: $\mathit{iparams}$, A_contr, B_contr
Task: find $S \in \{0,1\}^\ell$ such that $\mathrm{HW}(S \oplus S_A) \leq t$ and $\mathrm{HW}(S \oplus S_B) \leq t$.

Like the state recovery problems, the noisy key problem comes with a decisional variant. This problem captures the task of determining whether a candidate session key is close enough to Alice's and Bob's views.

Noisy Key Distinguishing (NKD).

Input: iparams, *A_contr*, *B_contr*, S; where if $b = 0$, $S \xleftarrow{\$} \{S \,|\, \textsc{hw}(S \oplus S_A) \leq t \wedge \textsc{hw}(S \oplus S_B) \leq t\}$, and if $b = 1$, $S \xleftarrow{\$} \{0,1\}^\ell$

Task: output 1 if $\textsc{hw}(S \oplus S_A) \leq t$ and $\textsc{hw}(S \oplus S_B) \leq t$; and 0 otherwise.

Clearly, a solver for ASR or for BSR can be used to solve NKS; and a solver for NKS can be used to solve NKD. Therefore, the strongest assumption associated to these problems is assuming that NKD is hard.

Assumption 1 (NKD assumption). *The given NKA protocol* $\Pi = (\mathsf{Init}, \mathsf{AContr}, \mathsf{BContr}, \mathsf{AConv}, \mathsf{BConv})$ *with noise level* t *and correctness error* ϵ *is such that for all polynomial time adversaries* A *in the* NKD *game (Game 4), their advantage* $\mathsf{Adv}^{\mathsf{NKD}}_{\Pi}(\mathsf{A})$ *is negligible:*

$$\mathsf{Adv}^{\mathsf{NKD}}_{\Pi}(\mathsf{A}) \triangleq \left| \Pr[\mathsf{Game}^{\mathsf{A}}_{\mathsf{NKD}}(1^\lambda) \not\Rightarrow 0] - \frac{1+\epsilon}{2} \right| \leq \mathsf{negl}(\lambda) \ . \tag{4}$$

When the argument is omitted, the expression denotes the maximum of this quantity across all quantum polynomial-time adversaries: $\mathsf{Adv}^{\mathsf{NKD}}_{\Pi} \triangleq \max_{\mathsf{A}} \mathsf{Adv}^{\mathsf{NKD}}_{\Pi}(\mathsf{A})$.

Game 4: $\mathsf{NKD}^{\mathsf{A}}(1^\lambda)$

1. $iparams \leftarrow \mathsf{Init}(1^\lambda)$
2. $A\_state, A\_contr \leftarrow \mathsf{AContr}(iparams)$
3. $B\_state, B\_contr \leftarrow \mathsf{BContr}(iparams)$
4. $S_A \leftarrow \mathsf{AConv}(A\_state, B\_contr)$
5. $S_B \leftarrow \mathsf{BConv}(B\_state, A\_contr)$
6. **if** $\textsc{hw}(S_A \oplus S_B) > t$ **then:**
7. ⌊ **return** $\perp$
8. $b \xleftarrow{\$} \{0,1\}$
9. **if** $b = 1$ **then:**
10. ⌊ $S \xleftarrow{\$} \{x \in \{0,1\}^\ell \,|\, \textsc{hw}(x \oplus S_A) \leq t \wedge \textsc{hw}(x \oplus S_B) \leq t\}$
11. **else:**
12. ⌊ $S \xleftarrow{\$} \{0,1\}^\ell$
13. $\hat{b} \leftarrow \mathsf{A}(iparams, A\_contr, B\_contr, S)$
14. **return** $[\![\hat{b} = b]\!]$

An interesting problem arises in the formalization of this assumption when the two parties' views of the session key is *too* different. In other words, whenever $\textsc{hw}(S_A \oplus S_B) > t$. Assumption 1 deals with this issue by aborting and ignoring the adversary in this case, but conservatively counting these events as wins for the adversary.

Whether or not to count these aborts as wins for the adversary is a matter of context. In one extreme, when a failure event occurs all bets are off in terms of security. In the other extreme, security is *only* compromised when the adversary successfully attacks a *successful* session. We choose the first option as it is more conservative and as the alternative implies complex design constraints. Note that an adversary whose strategy is random guess has success probability $\Pr[\mathsf{Game}^{\mathsf{A}}_{\mathsf{NKD}}(1^\lambda) \not\Rightarrow 0] = \epsilon + (1-\epsilon)\cdot\frac{1}{2} = \frac{1+\epsilon}{2}$ and hence advantage 0.

3.3 Security

We define the security of an NKA protocol in terms of the NKD game. This follows the regular Diffie-Hellman case in the *authenticated links model*, where security is based on the DDH assumption.

Definition 2 (security of NKA protocols). *An NKA protocol Π is* secure *if and only if the NKD Assumption holds for Π.*

So far, the identification of security with the NKD game has been justified by two arguments. First, the hardness of NKD implies the hardness of NKS, ASR, and BSR. Second, this identification mirrors the case of regular Diffie-Hellman. We supplement this justification with two more arguments. The next section studies a cryptosystem where ASR/BSR are hard, but which failed because NKD is not. Appendix B considers an alternate definition of security called *noisy key security (NK-security)*, along the lines of the session-key security (SK-security) notion in the authenticated links model of Canetti and Krawcyzk [20]. The conclusion of this appendix is that NK-security and the NKD Assumption are equivalent, up to a polynomial factor related to the number of sessions started and corrupted in the NK-security game. These indications strongly suggest that the NKD game is not merely a useful formalism, but an essential point of consideration in the context of noisy key agreement protocols.

3.4 Case Study: CFPKM

CFPKM [50] was a KEM proposal based on polynomial system solving with noise (PoSSoWN) submitted to the NIST project. Despite featuring a proof of security, the cryptosystem was broken within days. Since it implicitly relies on a noisy key agreement protocol, it is worthwhile to study what went wrong through the lens of the generic problems described above. The following description is simplified for clarity.

A CFPKM public key consists of a seed *seed* and a vector $\mathbf{b_1} \in \mathbb{Z}_q^m$, where *seed* is expanded into a list of m quadratic polynomials $\mathcal{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))$ with small coefficients in n variables $\mathbf{x} = (x_1, \ldots, x_n)$ over $\mathbb{Z}_q$ with q a power of 2. The secret key is a short vector $\mathbf{sa} \in \mathbb{Z}_q^n$ and the vector $\mathbf{b_1}$ is found as $\mathbf{b_1} = \mathcal{F}(\mathbf{sa}) + \mathbf{e_1}$ with $\mathbf{e_1} \in \mathbb{Z}_q^m$ a vector of small random errors. To encapsulate, the user chooses a random short vector $\mathbf{sb} \in \mathbb{Z}_q^n$. The ciphertext is then $\mathcal{F}(\mathbf{sb}) + \mathbf{e_2}$, where $\mathbf{e_2}$ is also a vector of small random errors, in addition to some

reconciliation information. The key is obtained as the most significant bits of $\mathbf{b_1} \odot \mathcal{F}(\mathbf{sb})$, where $\odot$ is the component-wise product. The decapsulator obtains the same key by computing $\mathcal{F}(\mathbf{sa}) \odot (\mathcal{F}(\mathbf{sb}) + \mathbf{e_2})$ and taking the most significant bits of this vector's components, and by correcting occasional errors when necessary. We identify the underlying noisy key agreement protocol with functionalities and noisy key views as follows. We use $\mathsf{msb}(\cdot)$ to denote the function that takes the most significant bits from each component of its vector argument.

Init:	generate $\mathcal{F}$ from *seed*
AContr:	sample $\mathbf{sa}, \mathbf{e_1}$ and transmit $\mathbf{b_1} = \mathcal{F}(\mathbf{sa}) + \mathbf{e_1}$
BContr:	sample $\mathbf{sb}, \mathbf{e_2}$ and transmit $\mathbf{b_2} = \mathcal{F}(\mathbf{sb}) + \mathbf{e_2}$
AConv:	compute $v_1 = \mathsf{msb}(\mathbf{b_2} \odot \mathcal{F}(\mathbf{sa}))$
BConv:	compute $v_2 = \mathsf{msb}(\mathbf{b_1} \odot \mathcal{F}(\mathbf{sb}))$
S_A:	v_1
S_B:	v_2

This description gives rise to the following instantiations of the abstract hard problems. The state recovery problems are instances of PoSSoWN.

A State Recovery (ASR).
Input: $\mathcal{F}, \mathbf{b_1}$ s.t. $\mathbf{b_1} = \mathcal{F}(\mathbf{sa}) + \mathbf{e_1}$ for some small $\mathbf{e_1}, \mathbf{sa}$
Task: find $\mathbf{sa}, \mathbf{e_1}$ s.t. $\mathbf{b_1} = \mathcal{F}(\mathbf{sa}) + \mathbf{e_1}$

B State Recovery (BSR).
Input: $\mathcal{F}, \mathbf{b_2}$ s.t. $\mathbf{b_2} = \mathcal{F}(\mathbf{sb}) + \mathbf{e_2}$ for some small $\mathbf{e_2}, \mathbf{sb}$
Task: find $\mathbf{sb}, \mathbf{e_2}$ s.t. $\mathbf{b_2} = \mathcal{F}(\mathbf{sb}) + \mathbf{e_2}$

Noisy Key Search (NKS).
Input: $\mathcal{F}, \mathbf{b_1}, \mathbf{b_2}$ such that $\mathbf{b_1} = \mathcal{F}(\mathbf{sa}) + \mathbf{e_1}$ and $\mathbf{b_2} = \mathcal{F}(\mathbf{sb}) + \mathbf{e_2}$ for some short $\mathbf{sa}, \mathbf{sb}, \mathbf{e_1}, \mathbf{e_2}$
Task: find $S \in \{0,1\}^\ell$ such that $\textsc{hw}(S \oplus v_1) \leq t$ and $\textsc{hw}(S \oplus v_2) \leq t$, where $v_1 = \mathsf{msb}(\mathcal{F}(\mathbf{sa}) \odot \mathbf{e_1})$ and $v_2 = \mathsf{msb}(\mathcal{F}(\mathbf{sb}) \odot \mathbf{e_2})$.

Noisy Key Distinguishing (NKD).
Input: $\mathcal{F}, \mathbf{b_1}, \mathbf{b_2}, S$ such that $\mathbf{b_1} = \mathcal{F}(\mathbf{sa}) + \mathbf{e_1}$ and $\mathbf{b_2} = \mathcal{F}(\mathbf{sb}) + \mathbf{e_2}$ for some short $\mathbf{sa}, \mathbf{sb}, \mathbf{e_1}, \mathbf{e_2}$
Task: decide whether $\textsc{hw}(S \oplus v_1) \leq t$ and $\textsc{hw}(S \oplus v_2) \leq t$, where $v_1 = \mathsf{msb}(\mathcal{F}(\mathbf{sa}) \odot \mathbf{e_1})$ and $v_2 = \mathsf{msb}(\mathcal{F}(\mathbf{sb}) \odot \mathbf{e_2})$.

The parameters of CFPKM are chosen to guarantee that the solution of the ASR/BSR problems have an infeasible target complexity. However, our analysis suggests that the hardness of ASR and BSR is not enough. Instead, one must look at NKD and tragically, it turns out that in this case NKD is not hard at all. In fact, the attack actually solves the NKS problem for a large proportion of instances by computing $v = \mathsf{msb}(\mathbf{b_1} \odot \mathbf{b_2})$.

Appendix A presents a similar analysis of several KEMs chosen as suitable representatives for their proper branches of mathematics, and identifies the induced hard problems and associated hardness assumptions. This demonstrates that our syntax and hard problems are generic and indeed capable of capturing a multitude of noisy key agreement based schemes. The examples treated there

are not known to be insecure. That is to say: there are no known attacks on the induced NKD problems.

4 NKA to KEM: Generic Construction

This section presents a transformation to obtain a KEM from a noisy key agreement protocol. In a nutshell, the public key is one contribution to the protocol. The random coins of the encapsulation algorithm are deterministically derived from its seed $s \in \{0,1\}^\lambda$ via a XOF. This algorithm generates the other protocol contribution and uses his view S_B of the shared noisy session key as a one-time pad to mask an encoding (using some error-correcting code) of the seed $s \in \{0,1\}^\lambda$. The decapsulation algorithm derives its own view S_A of the shared noisy session key to undo the one time pad up to some errors, after which it can decode the noisy codeword and obtain the seed s. At this point, the decapsulation algorithm simulates the encapsulation algorithm with the exact same deterministic parameters and verifies that the produced ciphertext is identical to the received one.

The resulting KEM is shown in Algorithms 5, 7, and 8. (For a syntactically correct presentation we split the probabilistic portion of the encapsulation from the deterministic portion.) The transformation's parameters are

- Π, the noisy key agreement protocol with session key length ℓ, noise level t, and correctness error ϵ;
- $\mathcal{C}$, the error-correcting coder and decoder for a $[n \leq \ell, k = \lambda, d > t]$-code;
- $\mathsf{H}_1, \mathsf{H}_2 : \{0,1\}^* \to \{0,1\}^\lambda$, hash functions;
- $\mathsf{H}_3 : \{0,1\}^\lambda \to \{0,1\}^*$, a cryptographically secure variable output length function whose output is long enough to derandomize any polynomial-time probabilistic algorithm; this may be instantiated by a XOF but we make abstraction of the output length.

We denote the resulting tuple of algorithms as $\mathcal{K} = \mathsf{SNOTP}(\Pi, \mathcal{C}, \mathsf{H}_3, \mathsf{H}_1, \mathsf{H}_2)$.

algorithm KeyGen
input: 1^λ — security parameter
output: sk — secret key
pk — public key

1: $iparams \leftarrow \Pi.\mathsf{Init}(1^\lambda)$
2: $A\_state, A\_contr \leftarrow \Pi.\mathsf{AContr}(iparams)$
3: $pk \leftarrow (iparams, A\_contr)$
4: $sk \leftarrow (A\_state, pk)$
5: **return** sk, pk

Algorithm 5: Key Generation of the KEM.

algorithm DetEncaps
input: $pk = (iparams, A\_contr)$ — public key
$s \in \{0,1\}^\lambda$ — random seed
output: k — symmetric key
c — ciphertext

1: $B\_state, B\_contr \leftarrow \Pi.\mathsf{BContr}(iparams; \mathsf{H}_3(s))$
2: $S_B \leftarrow \Pi.\mathsf{BConv}(B\_state, A\_contr)$
3: $e \leftarrow \mathcal{C}.\mathsf{encode}(s)$
4: $c \leftarrow (B\_contr, e \oplus S_B, \mathsf{H}_2(s))$
5: $k \leftarrow \mathsf{H}_1(pk \| s)$
6: **return** k, c

Algorithm 6: Deterministic encapsulation algorithm of the KEM.

algorithm Encaps
input: $pk = (iparams, A\_contr)$ — public key
output: k — symmetric key
c — ciphertext

1: $s \xleftarrow{\$} \{0,1\}^\lambda$
2: **return** $\mathsf{DetEncaps}(pk, s)$

Algorithm 7: Encapsulation algorithm of the KEM.

algorithm Decaps
input: $sk = (A\_state, pk)$ — secret key
input: $c = (B\_contr, E, h)$ — ciphertext
output: k — symmetric key if successful, or $\perp$ indicating failure

1: $S_A \leftarrow \Pi.\mathsf{AConv}(A\_state, B\_contr)$
2: $s \leftarrow \mathcal{C}.\mathsf{decode}(E \oplus S_A)$
3: **if** $s = \perp$ **or** $\mathsf{H}_2(s) \neq h$ **then:**
4: **return** $\perp$
5: **end**
6: $k, c' \leftarrow \mathsf{DetEncaps}(pk, s)$
7: **if** $c' \neq c$ **then:**
8: **return** $\perp$
9: **end**
10: **return** k

Algorithm 8: Decapsulation of the KEM.

Ramstake uses a slight variant of this transformation [50]. The change there is in line 5 of DetEncaps where k is computed as $k \leftarrow \mathsf{H}_1(pk \| coins)$ instead, where

coins $= \mathsf{H}_3(s)$, *i.e.*, the same coins with which $\mathsf{DetEncaps}$ was derandomized. It is clear that this change does not degrade security, for example by setting $\mathsf{H}_1(pk\|s) = \mathsf{H}'_1(pk\|\mathsf{H}_3(s))$.

4.1 Decapsulation Injectivity

Our construction actually achieves something in addition to IND-CCA security: *decapsulation injectivity*. In other words, for any given secret key sk, and for every key k there is (with overwhelming probability) at most one ciphertext c such that $\mathsf{Decaps}(sk, c) = k$. This might sound alarming at first, for instance because it is well known that a public key encryption scheme where every message maps onto one ciphertext cannot be IND-CPA secure, let alone IND-CCA secure.

However, the crucial distinction is that the ciphertexts of KEMs represent encapsulations of *uniformly random* keys. In contrast, PKEs must encrypt *arbitrary* messages, thus enabling the attacker to engineer repeat queries or another attack scenario that requires choosing precisely which messages to encrypt.

Decapsulation injectivity addresses benign malleability, which is the ability of an attacker to modify ciphertexts only if the encapsulated key remains intact. Schemes based on noisy key agreement are inherently resilient to noise and as a result, a ciphertext with added noise may still decapsulate correctly. Also, in some cases the mathematical objects on which the protocol relies, do not have a unique bit-level representation; in this case an adversary can switch representations to obtain a ciphertext that decapsulates to the same key. IND-CCA alone is not sufficient to preclude benign malleability or attacks exploiting it.

5 NKA to KEM: Security Analysis

5.1 Correctness

Theorem 1 (correctness). *Let Π be an NKA protocol with failure probability ϵ. The failure probability of the KEM $\mathcal{K} = \mathsf{SNOTP}(\Pi, \mathcal{C}, \mathsf{H}_3, \mathsf{H}_1, \mathsf{H}_2)$ is*

$$\Pr\left[k_c \neq k_d \;\middle|\; \begin{array}{l} sk, pk \leftarrow \mathsf{KeyGen}(1^\lambda) \\ k_e, c \leftarrow \mathsf{Encaps}(pk) \\ k_d \leftarrow \mathsf{Decaps}(sk, c) \end{array}\right] = \epsilon \ . \tag{5}$$

Proof. By construction, we have $pk = (iparams, A\_contr)$, $sk = (A\_state, pk)$ and $c = (B\_contr, S_B \oplus \mathcal{C}.\mathsf{encode}(s), \mathsf{H}_2(s))$, where c is deterministically generated from s and pk. Moreover, the encapsulator finds $k = \mathsf{H}_1(pk\|s)$. The decapsulator then computes $S_A = \Pi.\mathsf{AConv}(A\_state, B\_contr)$ and with probability ϵ, the strings S_A and S_B will lie too far apart for correct decoding. However, if $\mathrm{HW}(S_A \oplus S_B) \leq t$, then the decapsulator obtains the correct s from which he can produce the exact same ciphertext as well as $k = \mathsf{H}_1(pk\|s)$. In other words, there is a KEM decapsulation failure only when there is an NKA protocol failure. □

5.2 Techniques

We first explain some tools used in the proof before presenting the proof itself.

Inversion. The task of the simulator is to find a preimage x for an output image $y = \mathsf{H}(x)$ that was also output by the simulated algorithm H. In the classical random oracle model, the simulator B can peruse the list of queries made by A to H and test each such query x_i for $\mathsf{H}(x_i) \stackrel{?}{=} y$.

In the quantum random oracle model, this list of queries is ill-defined because the queries themselves may be represented by quantum superposition states. Instead, it is possible to accomplish the same thing by replacing the random function with a random polynomial $\mathsf{H} \in \mathbb{F}_{2^\ell}[x]$ of degree $2\hat{Q} - 1$, where $\hat{Q}$ is the number of queries made by A to H. Given the output image y, the simulator can factor $\mathsf{H}(x) - y$ in polynomial time to obtain the a list of at most $2\hat{Q}-1$ candidates $\{x_i\}_{i=0}^{2\hat{Q}-2}$. By selecting one at random, the simulator obtains the correct preimage with probability $\frac{1}{2\hat{Q}-1}$. Zhandry shows that $2\hat{Q}$-wise independent functions (such as this polynomial) are perfectly indistinguishable from a random function [67]. To the best of our knowledge, this technique for inversion in the quantum random oracle model was first used by Unruh for his non-interactivity transform [63].

Insecurity under One-Wayness. Recall that in the One-Wayness game, the challenger samples a random preimage x and runs the adversary on input $\mathsf{H}(x)$. The adversary wins if he outputs a y such that $\mathsf{H}(x) = y$. To capture the hardness of this task, we use a result by Unruh [63]. Here the adversary is given access to a random Bernoulli-distributed function $\mathsf{F} : \{0,1\}^* \to \{0,1\}$ and each $\mathsf{F}(x)$ is independently Bernoulli-distributed with $\Pr[\mathsf{F}(x) = 1] = \gamma$. For any quantum adversary A making at most $\hat{Q}$ queries, $\Pr[\mathsf{F}(\mathsf{A}^{\mathsf{F}}()) = 1] \leq 2(\hat{Q} + 1)\sqrt{\gamma}$. An adversary finding a preimage x of y in the One-Wayness game is simultaneously finding a preimage x of 1 for the Bernoulli-distributed function $\mathsf{F}(x) = [\![\mathsf{F}(x) = y]\!]$, and so $\Pr[\mathsf{H}(\mathsf{A}^{\mathsf{H}}(\mathsf{H}(x))) = \mathsf{H}(x)] \leq 2(\hat{Q}_{\mathsf{H}} + 1)\sqrt{2^{-n}}$, where n is the output length of H.

Aggregate quantum query amplitude. Our proof relies in part on the indistinguishability of two worlds predicated on a certain value s not being queried to the random oracle. Classically, we can define $b_{k,s} \in \{0,1\}$ as the Boolean value that takes the value 1 in the worlds where the value of query k is s, and 0 in the worlds where it is not, and then proceed to make a distinction depending on whether the aggregation $a_s = \bigvee_k b_{k,s}$ equals 1. In the quantum case, however, these variables are ill-defined because each query does not have an associated value but an associated quantum state, which might be a superposition of many values with possibly non-uniform amplitudes. Nevertheless, we show that the argument can be made to work (even in the quantum random oracle model) when we look instead at these variables' expectation value $\mathrm{E}[b_{k,s}] \in \mathbb{R}_{\geq 0}$. To this end, we define the *quantum query amplitude* $\hat{b}_{k,s} \in \mathbb{C}$ at the kth query associated

with a set $\mathcal{S}$ of potential values, and its aggregate across all queries $\hat{a}_s$, in a way that mirrors (but does not capture) the classical notion.

Definition 3 (aggregate quantum query amplitude). *Let A^{H} be a quantum algorithm with oracle access to H making $\hat{Q}$ queries. In particular, A consists of $\hat{Q}+1$ unitary transforms $U_0, \ldots, U_{\hat{Q}}$ operating on a triple of quantum registers S, Q, R, and interleaved with unitaries H operating only on Q, R and sending $|q, r\rangle \mapsto |q, r \oplus \mathsf{H}(q)\rangle$. Let ρ_k^Q represent the reduced density matrix with respect to Q immediately after query k, with query indexation starting at zero. Then the* aggregate quantum query amplitude *$\hat{a}_{\mathcal{S}}$ associated with a set $\mathcal{S}$ of potential queries is*

$$\hat{a}_{\mathcal{S}} = \sum_{k=0}^{\hat{Q}-1} \sqrt{\sum_{s \in \mathcal{S}} \langle s|\rho_k^Q|s\rangle} \ . \tag{6}$$

The aggregate quantum query amplitude is useful as a standalone concept because it enables arguments that consider the degree to which an adversary is querying some member of a set $\mathcal{S}$ and how this quantity changes as this set is modified. The following two lemmas illustrate this fact.

Lemma 1. *For any two sets $\mathcal{S}_1, \mathcal{S}_2 \subseteq \{0,1\}^*$, $\hat{a}_{\mathcal{S}_1} \leq \hat{a}_{\mathcal{S}_1 \cup \mathcal{S}_2}$.*

Proof. Since $\langle s|\rho_k^Q|s\rangle$ is a positive quantity for any s, increasing the range of the sum from $\mathcal{S}_1$ to $\mathcal{S}_1 \cup \mathcal{S}_2$ can only make the sum larger. □

Lemma 2. *For any two sets $\mathcal{S}_1, \mathcal{S}_2 \subset \{0,1\}^*$, if $\hat{a}_{\mathcal{S}_1} \leq 1$ and $\hat{a}_{\mathcal{S}_2} \leq 1$ then $\hat{a}_{\mathcal{S}_1 \cup \mathcal{S}_2} \leq \hat{a}_{\mathcal{S}_1} + \hat{a}_{\mathcal{S}_2}$.*

Proof. Overload "\" such that $\mathcal{S}_2 \backslash \mathcal{S}_1 \triangleq \mathcal{S}_2 \backslash (\mathcal{S}_2 \cap \mathcal{S}_1)$. Then we have

$$\hat{a}_{\mathcal{S}_1 \cup \mathcal{S}_2} = \sum_{k=0}^{\hat{Q}-1} \sqrt{\sum_{s \in \mathcal{S}_1 \cup \mathcal{S}_2} \langle s|\rho_k^Q|s\rangle} = \sum_{k=0}^{\hat{Q}-1} \sqrt{\sum_{s \in \mathcal{S}_1} \langle s|\rho_k^Q|s\rangle + \sum_{s \in \mathcal{S}_2 \backslash \mathcal{S}_1} \langle s|\rho_k^Q|s\rangle} \tag{7}$$

$$\leq \sum_{k=0}^{\hat{Q}-1} \sqrt{\sum_{s \in \mathcal{S}_1} \langle s|\rho_k^Q|s\rangle} + \sum_{k=0}^{\hat{Q}-1} \sqrt{\sum_{s \in \mathcal{S}_2 \backslash \mathcal{S}_1} \langle s|\rho_k^Q|s\rangle} = \hat{a}_{\mathcal{S}_1} + \hat{a}_{\mathcal{S}_2 \backslash \mathcal{S}_1} \tag{8}$$

$$\leq \hat{a}_{\mathcal{S}_1} + \hat{a}_{\mathcal{S}_2} \ . \tag{9}$$

The first inequality holds because the terms in the square root are smaller than 1 because $\hat{a}_{\mathcal{S}_1} \leq 1$ and $\hat{a}_{\mathcal{S}_2 \backslash \mathcal{S}_1} \leq \hat{a}_{\mathcal{S}_2} \leq 1$. The second holds due to lemma 1. □

We now upper-bound the trace distance of any pair of quantum distinguishers $\mathsf{D}^{\mathsf{H}_b}$ with oracle access to H_b for some $b \in \{0,1\}$, where $\mathsf{H}_0(x) \neq \mathsf{H}_1(x) \implies x \in \mathcal{S}$, in terms of $\hat{a}_{\mathcal{S}}$. This trace distance in turn upper bounds the maximum distinguishing advantage across all adversaries. The following proof draws in large part on [8, Lemma 37].

Lemma 3. *Let* D *be a quantum distinguisher making at most* $\hat{Q}$ *queries to one of two oracles* $\mathsf{H}_0, \mathsf{H}_1$*, whose outputs differ only on a set* $\mathcal{S}$ *of inputs. Then the trace distance of the distinguishers' final states is bounded by*

$$\mathrm{TD}(\mathsf{D}^{\mathsf{H}_1}(), \mathsf{D}^{\mathsf{H}_2}()) \leq 2\hat{a}_{\mathcal{S}} \ . \tag{10}$$

Proof. Without loss of generality, D uses three registers S, Q, R for its state, and consists of unitary transformations $\{U_k\}_{k=0}^{\hat{Q}}$ operating on all three registers interleaved with oracle queries, which are also unitary transformations H_b but which operate only on Q, R and map $|q, r\rangle \mapsto |q, r \oplus \mathsf{H}_b(q)\rangle$. So if $|\Psi_0\rangle$ is the adversary's initial state, then its final state is given by $|\Psi_b^{\hat{Q}}\rangle = \left(\prod_{k=0}^{\hat{Q}-1} U_{\hat{Q}-k} H_b\right) |\Psi_0\rangle$.

Let $|\Psi_b^i\rangle = \left(\prod_{k=0}^{i} U_{i-k} H_b\right) |\Psi_0\rangle$ be the state before query number i (with indexation of queries starting at 0), and let $|\Psi_b^{\hat{Q}}\rangle$ denote the final state. Define the trace distance at stage i as

$$D_i = \mathrm{TD}(|\Psi_0^i\rangle, |\Psi_1^i\rangle) \ . \tag{11}$$

And then

$$\begin{aligned}
D_i &= \mathrm{TD}(|\Psi_0^i\rangle, |\Psi_1^i\rangle) && (12)\\
&= \mathrm{TD}(U_i H_0 |\Psi_0^{i-1}\rangle, U_i H_1 |\Psi_1^{i-1}\rangle) && (13)\\
&= \mathrm{TD}(H_0 |\Psi_0^{i-1}\rangle, H_1 |\Psi_1^{i-1}\rangle) && (14)\\
&\leq \mathrm{TD}(H_0 |\Psi_0^{i-1}\rangle, H_1 |\Psi_0^{i-1}\rangle) + \mathrm{TD}(H_1 |\Psi_0^{i-1}\rangle, H_1 |\Psi_1^{i-1}\rangle) && (15)\\
&= \mathrm{TD}(H_0 |\Psi_0^{i-1}\rangle, H_1 |\Psi_0^{i-1}\rangle) + \mathrm{TD}(|\Psi_0^{i-1}\rangle, |\Psi_1^{i-1}\rangle) && (16)\\
&= \mathrm{TD}(H_0 |\Psi_0^{i-1}\rangle, H_1 |\Psi_0^{i-1}\rangle) + D_{i-1} \ , && (17)
\end{aligned}$$

where the triangle inequality is used (15). Moreover, since $D_0 = 0$, we have

$$\mathrm{TD}(\mathsf{D}^{\mathsf{H}_0}(), \mathsf{D}^{\mathsf{H}_1}()) = D_{\hat{Q}} \leq \sum_{i=0}^{\hat{Q}-1} \mathrm{TD}(H_0 |\Psi_0^i\rangle, H_1 |\Psi_0^i\rangle) \ . \tag{18}$$

Now consider the projection operator $P_{\mathcal{S}}$ which operates on Q and projects onto the span of all $|s\rangle$ where $s \in \mathcal{S}$. Formally, $P_{\mathcal{S}} = \sum_{s \in \mathcal{S}} I_{(S)} \otimes |s\rangle\langle s| \otimes I_{(R)}$. Let $P_{\bar{\mathcal{S}}}$ be its complement, *i.e.*, $P_{\bar{\mathcal{S}}} = \sum_{s \notin \mathcal{S}} I_{(S)} \otimes |s\rangle\langle s| \otimes I_{(R)}$. We use the symbol z to represent values contained in register S; r for values in R; and both q and s for values in Q.

$$\mathrm{TD}(H_0|\Psi_0^i\rangle, H_1|\Psi_0^i\rangle) = \mathrm{TD}((P_{\mathcal{S}} + P_{\bar{\mathcal{S}}})H_0|\Psi_0^i\rangle, (P_{\mathcal{S}} + P_{\bar{\mathcal{S}}})H_1|\Psi_0^i\rangle) \tag{19}$$

$$= \mathrm{TD}(P_{\mathcal{S}}H_0|\Psi_0^i\rangle + P_{\bar{\mathcal{S}}}H_0|\Psi_0^i\rangle, P_{\mathcal{S}}H_1|\Psi_0^i\rangle + P_{\bar{\mathcal{S}}}H_1|\Psi_0^i\rangle) \tag{20}$$

$$= \mathrm{TD}(P_{\mathcal{S}}H_0|\Psi_0^i\rangle + P_{\bar{\mathcal{S}}}H_0|\Psi_0^i\rangle, P_{\mathcal{S}}H_1|\Psi_0^i\rangle + P_{\bar{\mathcal{S}}}H_0|\Psi_0^i\rangle) \tag{21}$$

$$\leq 2\|P_{\mathcal{S}}H_0|\Psi_0^i\rangle\| \tag{22}$$

$$= 2\sqrt{\langle\Psi_0^i|H_0^\dagger P_{\mathcal{S}}^\dagger P_{\mathcal{S}}H_0|\Psi_0^i\rangle} \tag{23}$$

$$= 2\sqrt{\sum_{s\in\mathcal{S}}\langle\Psi_0^i|H_0^\dagger(I_{(S)}\otimes|s\rangle\langle s|\otimes I_{(R)})H_0|\Psi_0^i\rangle} \tag{24}$$

$$= 2\sqrt{\sum_{s\in\mathcal{S}}\sum_{z,r}(\langle z|\otimes\langle s|\otimes\langle r|)H_0|\Psi_0^i\rangle\langle\Psi_0^i|H_0^\dagger(|z\rangle\otimes|s\rangle\otimes|r\rangle)} \tag{25}$$

$$= 2\sqrt{\sum_{s\in\mathcal{S}}\langle s|\rho_i^Q|s\rangle}\ . \tag{26}$$

Equation 21 holds because H_0 and H_1 are only different when $q \in \mathcal{S}$, so their effect is the same when projecting onto $\mathsf{span}(\{|s\rangle\}_{s\notin\mathcal{S}})$. The inequality (22) holds due to [8, lemma 35] (with $|\Phi\rangle = P_{\bar{\mathcal{S}}}H_0|\Psi_0^i\rangle$). Equation 26 holds because the reduced density operator of $H_0|\Psi_0^i\rangle = \sum_{z,q,r}\alpha_{z,q,r}|z,q,r\oplus\mathsf{H}_0(q)\rangle$ with respect to register Q is given by

$$\rho_i^Q = \mathsf{Tr}_{S,R}\left(H_0|\Psi_0^i\rangle\langle\Psi_0^i|H_0^\dagger\right) \tag{27}$$

$$= \sum_{z_1,z_2}\sum_{q_1,q_2}\sum_{r_1,r_2}\alpha_{z_1,q_1,r_1}\alpha^\dagger_{z_2,q_2,r_2}\langle z_1|z_2\rangle\langle r_1\oplus\mathsf{H}_0(q_1)|r_2\oplus\mathsf{H}_0(q_2)\rangle|q_1\rangle\langle q_2| \tag{28}$$

$$= \sum_{z}\sum_{q_1,q_2}\sum_{r_1,r_2}\alpha_{z,q_1,r_1}\alpha^\dagger_{z,q_2,r_2}\langle z|z\rangle\langle r_1\oplus\mathsf{H}_0(q_1)|r_2\oplus\mathsf{H}_0(q_2)\rangle|q_1\rangle\langle q_2| \tag{29}$$

$$= \sum_{z}\sum_{q_1,q_2}\left(\sum_{r_1,r_2\,|\,r_1\oplus\mathsf{H}_0(q_1)=r_2\oplus\mathsf{H}_0(q_2)}\alpha_{z,q_1,r_1}\alpha^\dagger_{z,q_2,r_2}\right)|q_1\rangle\langle q_2| \tag{30}$$

$$= \sum_{z}\sum_{q_1,q_2}\sum_{r}\alpha_{z,q_1,r\oplus\mathsf{H}_0(q_1)}\alpha^\dagger_{z,q_2,r\oplus\mathsf{H}_0(q_2)}|q_1\rangle\langle q_2|\ . \tag{31}$$

In particular, this means that

$$\sum_{s\in\mathcal{S}}\langle s|\rho_i^Q|s\rangle = \sum_{s\in\mathcal{S}}\langle s|\left(\sum_z\sum_{q_1,q_2}\sum_r \alpha_{z,q_1,r\oplus\mathsf{H}_0(q_1)}\alpha^\dagger_{z,q_2,r\oplus\mathsf{H}_0(q_2)}|q_1\rangle\langle q_2|\right)|s\rangle \quad (32)$$

$$= \sum_{s\in\mathcal{S}}\left(\sum_z\sum_{q_1,q_2}\sum_r \alpha_{z,q_1,r\oplus\mathsf{H}_0(q_1)}\alpha^\dagger_{z,q_2,r\oplus\mathsf{H}_0(q_2)}\langle s|q_1\rangle\langle q_2|s\rangle\right) \quad (33)$$

$$= \sum_{s\in\mathcal{S}}\sum_z\sum_r \alpha_{z,s,r\oplus\mathsf{H}_0(s)}\alpha^\dagger_{z,s,r\oplus\mathsf{H}_0(s)} \quad (34)$$

$$= \sum_{s\in\mathcal{S}}\sum_z\sum_r \alpha_{z,s,r\oplus\mathsf{H}_0(s)}\alpha^\dagger_{z,s,r\oplus\mathsf{H}_0(s)}\left(\langle z|\otimes\langle s|\otimes\langle r\oplus\mathsf{H}_0(s)|\right)\left(|z\rangle\otimes|s\rangle\otimes|r\otimes\mathsf{H}_0(s)\rangle\right) \quad (35)$$

$$= \sum_{s\in\mathcal{S}}\sum_z\sum_r \left(\langle z|\otimes\langle s|\otimes\langle r\oplus\mathsf{H}_0(s)|\right)H_0|\Psi_0^i\rangle\langle\Psi_0^i|H_0^\dagger\left(|z\rangle\otimes|s\rangle\otimes|r\otimes\mathsf{H}_0(s)\rangle\right) \ . \quad (36)$$

Consequently,

$$\mathrm{TD}(\mathsf{D}^{\mathsf{H}_0}(),\mathsf{D}^{\mathsf{H}_1}()) = D_{\hat{Q}} \le 2\sum_{k=0}^{\hat{Q}-1}\sqrt{\sum_{s\in\mathcal{S}}\langle s|\rho_k^Q|s\rangle} = 2\hat{a}_{\mathcal{S}} \ . \qquad \square \quad (37)$$

This theorem shows that if an algorithm A is capable of making a distinction between H_0 and H_1, where H_0 and H_1 differ only on a set $\mathcal{S}$, then $\hat{a}_{\mathcal{S}}$ must be large. The next lemma completes the reasoning by lower-bounding the success probability of an extractor machine who, given black-box access to A, H_0, and H_1, attempts to output some $s \in \mathcal{S}$.

Lemma 4 (Multi-target one-way to hiding). *Let H_0 and H_1 be oracle functions that differ only on input set $\mathcal{S}$, and let A be a quantum adversary that makes at most $\hat{Q}_{\mathsf{H}}$ queries to either H_0 or H_1. Let E be the following algorithm: select $b \xleftarrow{\$} \{0,1\}$ and $k \xleftarrow{\$} \{0,\ldots,\hat{Q}_{\mathsf{H}}-1\}$ at random, simulate $\mathsf{A}^{\mathsf{H}_b}$ until the kth query, measure the query register in the computational basis, and output the result. Then*

$$\Pr[\mathsf{E}^{\mathsf{A},\mathsf{H}_0,\mathsf{H}_1}() \Rightarrow s\in\mathcal{S}] \ge \left(\frac{\hat{a}_{\mathcal{S}}}{\hat{Q}_{\mathsf{H}}}\right)^2 \ge \left(\frac{1}{2\hat{Q}_{\mathsf{H}}}\mathrm{TD}(\mathsf{A}^{\mathsf{H}_0}(),\mathsf{A}^{\mathsf{H}_1}())\right)^2 \ . \quad (38)$$

Proof. The probability that E outputs a member of $\mathcal{S}$ is given by

$$\Pr[\mathsf{E}^{\mathsf{A},\mathsf{H}_0,\mathsf{H}_1}() \Rightarrow s\in\mathcal{S}] = \sum_{k=0}^{\hat{Q}_{\mathsf{H}}-1}\sum_{s\in\mathcal{S}}\Pr[\mathsf{E}^{\mathsf{A},\mathsf{H}_0,\mathsf{H}_1}() \Rightarrow s \wedge \mathsf{E}\ \textit{chooses}\ k] \quad (39)$$

$$= \sum_{k=0}^{\hat{Q}_{\mathsf{H}}-1}\sum_{s\in\mathcal{S}}\langle s|\rho_k^Q|s\rangle\cdot\frac{1}{\hat{Q}_{\mathsf{H}}} \ . \quad (40)$$

Compare with $\hat{a}_{\mathcal{S}}$, which is bounded by via Jensen's inequality by

$$\hat{a}_{\mathcal{S}} = \sum_{k=0}^{\hat{Q}_{\mathsf{H}}-1} \sqrt{\sum_{s\in\mathcal{S}} \langle s|\rho_k^Q|s\rangle} = \hat{Q}_{\mathsf{H}} \sum_{k=0}^{\hat{Q}_{\mathsf{H}}-1} \frac{1}{\hat{Q}_{\mathsf{H}}} \sqrt{\sum_{s\in\mathcal{S}} \langle s|\rho_k^Q|s\rangle} \tag{41}$$

$$\leq \hat{Q}_{\mathsf{H}} \sqrt{\sum_{k=0}^{\hat{Q}_{\mathsf{H}}-1} \frac{1}{\hat{Q}_{\mathsf{H}}} \sum_{s\in\mathcal{S}} \langle s|\rho_k^Q|s\rangle} \ . \tag{42}$$

Plugging in Eqn. 40 and Lemma 3 yields the theorem statement. □

We draw attention to some differences with respect to Unruh's one-way to hiding lemma [64]. First, our lemma works with an arbitrary potential query set $\mathcal{S}$, whereas Unruh's lemma works only for a single query. Second, our lemma does not assume H_0 and H_1 are random functions per se, but only that they are black boxes accessed as oracles. Third, in Unruh's lemma the adversary A has access to only one random oracle H and his input is the query-response pair (x, z), where either $z = \mathsf{H}(x)$ or $z = y \neq \mathsf{H}(x)$, and his task is to decide which is the case. In our lemma the distinguisher D is tasked with distinguishing which of two different oracles he has access to. This difference is immaterial, however, since one can used to derive the other. In fact, Unruh's original proof starts by translating the problem into distinguishing two oracles that differ only on x.

5.3 Security Reduction

The security bound involves three parameters determined by the NKA protocol: ϵ and ϕ. The first is the failure probability. The second warrants some explanation. In the NKD game when $b = 0$, S is sampled uniformly at random. However, there is a small probability that this uniform S happens to lie in the radius-t sphere centered at S_A, and in this case the adversary might decide that the ciphertext is correctly formed or decapsulate it outright and indicate incorrectly that $b = 1$. We therefore capture this probability explicitly: $\phi = \left(\sum_{k=0}^{t} \binom{\ell}{k}\right) / 2^{\ell}$.

The construction involves two hash functions, H_1 and H_2, and one variable output function, H_3. In the security argument these are modeled as random oracles.

Theorem 2 (IND-CCA security if NKD Assumption holds). *Let* A *be a quantum adversary in the* IND-CCA *game against* $\mathcal{K} = \mathsf{SNOTP}(\Pi, \mathcal{C}, \mathsf{H}_3, \mathsf{H}_1, \mathsf{H}_2)$. *Let and* $Q_d, \hat{Q}_{\mathsf{H}_1}, \hat{Q}_{\mathsf{H}_2}, \hat{Q}_{\mathsf{H}_3}$ *its number of queries to the decapsulation oracle,* H_1, H_2 *and* H_3, *respectively. Let* ℓ, t *and* ϵ *be the session key length, noise threshold, and failure probability of the NKA protocol* Π. *Then the advantage* $\mathsf{Adv}_{\mathcal{K}}^{\mathsf{IND\text{-}CCA}}(\mathsf{A})$ *is upper bounded by*

$$\mathsf{Adv}_{\mathcal{K}}^{\mathsf{IND\text{-}CCA}}(\mathsf{A}) \leq \frac{2\epsilon + \phi - \epsilon\phi}{2(1-\phi)(1-\epsilon)} + \frac{3 - 2\epsilon - 2\phi + 2\epsilon\phi}{(1-\epsilon)(1-\phi)} \mathsf{Adv}_{\Pi}^{\mathsf{NKD}} \tag{43}$$

$$+ 2\hat{Q}_{\mathsf{H}_3} \sqrt{2(\hat{Q}_{\mathsf{H}_1} + 1)\sqrt{2^{-\lambda}}} + 2\hat{Q}_{\mathsf{H}_3} \sqrt{2(\hat{Q}_{\mathsf{H}_2} + 1)\sqrt{2^{-\lambda}}} + 4\hat{Q}_{\mathsf{H}_3} \sqrt{\mathsf{Adv}_{\Pi}^{\mathsf{NKD}}}$$

in the quantum random oracle model.

Proof. The proof follows from a sequence of games arguments. At each iteration, a simulator is simulating the previous game and the previous game's adversary in order to win the next game.

– Game 1 is identical to the IND-CCA for KEMs game against $\mathcal{K}$. So by definition,

$$\Pr[\mathsf{Game}\,1^{\mathsf{A}^{\mathsf{D}(\cdot)}}(1^\lambda) \Rightarrow 1] = \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathcal{K}}(\mathsf{A}) \ . \tag{44}$$

– Game 2 is the IND-CPA game against a variant of the KEM that drops derandomization. In particular, there are three modifications: a) the modified algorithm $\mathsf{DetEncaps}'$ is identical to DetEncaps except for line 1, which becomes

$$1\text{: } B\_state,\ B\_contr \leftarrow \Pi.\mathsf{BContr}(iparams)\ ;$$

b) Decaps' is identical to Decaps except with lines 6—9 replaced by

$$6\text{: } k \leftarrow \mathsf{H}_1(pk\|s)\ ;$$

c) the hash value h is dropped from the ciphertext and line 3 of Decaps becomes

$$3\text{: } \textbf{if } s = \perp \textbf{ then:} \quad .$$

The adversary B of Game 2 simulates A and is therefore responsible for making A's view of events as close to an authentic run of Game 1 as possible. In particular, B forwards all queries to the oracles to its oracles $\mathsf{H}_1, \mathsf{H}_3$ and forwards all responses back. However, B presents A with a backdoored random oracle H_2 which is really a random polynomial of degree at most $2\hat{Q}_{\mathsf{H}_2} - 1$. The purpose of this switch is to be able to answer decapsulation queries as follows.

```
1. define D(q) as:
2.     B_contr, E, h ← q
3.     factors ← factorize(H_2(x) − h)
4.     for x ∈ factors do:
5.         k', c' ← DetEncaps(pk, x)
6.         if c' = q then return k'
7.     return ⊥
```

Since H_2 is a $2\hat{Q}_{\mathsf{H}_2}$-wise independent function, it is perfectly indistinguishable from a true random oracle as long as at most $\hat{Q}_{\mathsf{H}_2}$ queries are made to it. Consequently, this simulated random oracle does not affect security or winning probability. The simulator's running time does increase as a result of this inversion strategy. For every query, he has to factorize a degree at most $2\hat{Q}_{\mathsf{H}_2} - 1$ polynomial and then for each of the at most $2\hat{Q}_{\mathsf{H}_2} - 1$ factors run the deterministic encapsulation procedure followed by some testing. Nevertheless, this operational cost is still linear in Q_d and polynomial in $\hat{Q}_{\mathsf{H}_2}$.

At some point, the simulator B receives the challenge ciphertext-key pair (c, k), where c is lacking a hash-of-seed h. The simulator appends a random value $h^* \xleftarrow{\$} \{0,1\}^\lambda$ to the ciphertext before forwarding it, along with the challenge key, to the adversary A. The simulator B outputs whatever the adversary A outputs.

The difference in input distribution of A when it is playing $\mathsf{Game}\ 1$ versus when it is being simulated by B is characterized by the fact that no $s' \in \{0,1\}^\lambda$ satisfies $\mathsf{DetEncaps}'(pk, \mathsf{H}_3(s')) = (c, \cdot)$ in the latter case. Therefore, provided that A fails to query H_3 on likely candidates for s, the difference in winning probability of A and B in their proper games is negligible. To formalize this argument, consider the adversary's aggregate quantum query amplitude $\hat{a}_{\mathcal{S}}$ on H_3 for the set $\mathcal{S}$ whose members s' satisfy:

- $\mathsf{H}_1(pk \| s') = k$, or
- $\mathsf{H}_2(s') = h^*$, or
- $\Pi.\mathsf{BContr}(iparams; \mathsf{H}_3(s')) = (B\_contr, \cdot)$, or
- $\Pi.\mathsf{BContr}(iparams; \mathsf{H}_3(s')) = (\cdot, B\_state)$ and $\mathcal{C}.\mathsf{decode}(\Pi.\mathsf{BConv}(A\_contr, B\_state) \oplus E) = s'$, or
- $s' = s$.

This list is exhaustive because any s' that does not satisfy any of these conditions is independent of the provided ciphertext and key. The first bullet point represents $\mathsf{H}_1(pk, \cdot)^{-1}$, the set of preimages of k under $\mathsf{H}_1(pk, \cdot)$. The second bullet point represents $\mathsf{H}_2^{-1}(k)$, the set of preimages of h^* under H_2. The next two bullet points represent the set $\mathcal{S}_{\mathsf{H}_3}$, the set of preimages under H_3 to bitstrings that, when fed as random tape to $\Pi.\mathsf{BContr}$, generate a state B_state or contribution B_contr with which the NKD game is won. The last bullet point indicates that the adversary is querying the payload s, which he obtained from solving the NKS problem to find S and then decoding $S \oplus S_B \oplus \mathcal{C}.\mathsf{encode}(s)$.

By separating the aggregate amplitude along these lines we obtain using lemma 2

$$\hat{a}_{\mathcal{S}} \leq \hat{a}_{\mathsf{H}_1(pk,\cdot)^{-1}(k)} + \hat{a}_{\mathsf{H}_2^{-1}(h^*)} + \hat{a}_{\mathcal{S}_{\mathsf{H}_3}} + \hat{a}_s \ . \tag{45}$$

The first two terms in this expression can be bounded by the an extractor's success probability at winning a One-Wayness game using lemma 4. Specifically,

$$\left(\frac{\hat{a}_{\mathsf{H}_1(pk,\cdot)^{-1}(k)}}{\hat{Q}_{\mathsf{H}_3}}\right)^2 \leq \Pr[\mathsf{E}^{\mathsf{A}}() \Rightarrow s \in \mathsf{H}_1(pk, \cdot)^{-1}(k)] \tag{46}$$

$$\leq 2(\hat{Q}_{\mathsf{H}_1} + 1)\sqrt{2^{-\lambda}} \ , \tag{47}$$

and similarly, $\hat{a}^2_{\mathsf{H}_2^{-1}(h^*)} \leq 2\hat{Q}^2_{\mathsf{H}_3}(\hat{Q}_{\mathsf{H}_2} + 1)\sqrt{2^{-\lambda}}$. With respect to the third term, observe that this gives rise to an extractor machine that solves NKD, so $\hat{a}_{\mathcal{S}_{\mathsf{H}_3}} \leq \hat{Q}_{\mathsf{H}_3}\sqrt{\mathsf{Adv}^{\mathsf{NKD}}_{\Pi}}$. The same is true for the fourth term but in a

roundabout manner. Define this fourth extractor machine as follows: E_4 takes an NKD instance $(iparams, A\_contr, B\_contr, S)$ and embeds this instance into a public key and ciphertext in order to simulate the adversary. In particular, the public key is $(iparams, A\_contr)$ and the ciphertext is $(B\_contr, \mathcal{C}.\mathsf{encode}(s) \oplus S, h)$ for randomly chosen s, h. Next, E_4 measures a random query to H_3 in the computational basis and outputs 1 if this measurement yields s and 0 otherwise. If the adversary solves NKS and queries s, then E_4 has a $1/\hat{Q}_{\mathsf{H}_3}$ chance of winning the NKD game. So

$$\hat{a}_s \leq \hat{Q}_{\mathsf{H}_3} \sqrt{\Pr[\mathsf{E}_4 \textit{ wins } \mathsf{NKD}]} \leq \hat{Q}_{\mathsf{H}_3} \sqrt{\mathsf{Adv}^{\mathsf{NKD}}_{\Pi}} \quad . \tag{48}$$

Putting these terms together we obtain

$$\hat{a}_{\mathcal{S}} \leq \hat{Q}_{\mathsf{H}_3} \left(\sqrt{2(\hat{Q}_{\mathsf{H}_1} + 1)\sqrt{2^{-\lambda}}} + \sqrt{2(\hat{Q}_{\mathsf{H}_2} + 1)\sqrt{2^{-\lambda}}} + 2\sqrt{\mathsf{Adv}^{\mathsf{NKD}}_{\Pi}} \right) \quad . \tag{49}$$

Without loss of generality, the behavior of H_1, H_2 and H_3 with respect to inputs $s' \notin \mathcal{S}$ is identical across games 1 and 2; in other words, these functions are only different on members of $\mathcal{S}$. This means that the distinguishing advantage of any adversary A across game 1 and game 2 (where it is being simulated by B) can be bounded using lemma 3:

$$|\Pr[\mathsf{Game}\,1^{\mathsf{A}}(1^\lambda) \Rightarrow 1] - \Pr[\mathsf{Game}\,2^{\mathsf{B}^{\mathsf{A}}}(1^\lambda) \Rightarrow 1]| \leq 2\hat{a}_{\mathcal{S}} \tag{50}$$

$$\leq 2\hat{Q}_{\mathsf{H}_3} \left(\sqrt{2(\hat{Q}_{\mathsf{H}_1} + 1)\sqrt{2^{-\lambda}}} + \sqrt{2(\hat{Q}_{\mathsf{H}_2} + 1)\sqrt{2^{-\lambda}}} + 2\sqrt{\mathsf{Adv}^{\mathsf{NKD}}_{\Pi}} \right) \quad . \tag{51}$$

– $\mathsf{Game}\ 3$ is the NKD game. The adversary C in this game simulates B and is thus responsible for making B's view of events as close as possible to an authentic execution of $\mathsf{Game}\ 2$. In particular, C uses its input as well as the challenge session key to generate the public key and a challenge ciphertext that transmits a random seed $s \xleftarrow{\$} \{0,1\}^\lambda$. He presents the simulated algorithm B with a random oracle H_1 that is programmed to output $k = \mathsf{H}_1(pk \| s)$ for some randomly chosen $k \xleftarrow{\$} \{0,1\}^\lambda$. At some point the simulated adversary B outputs a bit $\hat{b}$ and the simulator C outputs this same bit.
If an NKA failure event F occurs, then the simulator C "wins" regardless of the behavior of the adversary B — because its output $\perp$ contributes to the adversary's advantage just as much as the output 1.
If the adversary B wins with output $\hat{b} = 0$, then the ciphertext $c = (B\_contr, S \oplus \mathcal{C}.\mathsf{encode}(s))$ is not an encapsulation of k and consequently $\mathcal{C}.\mathsf{decode}(S_A \oplus S \oplus \mathcal{C}.\mathsf{encode}(s)) \neq s$. This implies that S_A and S are more than t bits apart, implying that S was chosen randomly because $b = 0$. So the simulator C wins by outputting $\hat{b}$.
If the adversary B wins with output $\hat{b} = 1$, then the ciphertext $c = (B\_contr, S \oplus \mathcal{C}.\mathsf{encode}(s))$ is an encapsulation of $k = \mathsf{H}_1(pk \| s)$, meaning that $\mathcal{C}.\mathsf{decode}(S_A \oplus S \oplus \mathcal{C}.\mathsf{encode}(s)) = s$. This implies that S_A is t or fewer bits apart from

S. This in turn implies one of two things; either that S was chosen from the intersection of spheres centered at S_A or S_B because $b = 1$; or else that S was drawn uniformly at random and happens to lie close to S_A. In the former case, the simulator who outputs $\hat{b} = 1$ wins as well. If $b = 0$, the latter case occurs with a probability $\phi = \left(\sum_{k=0}^{t} \binom{\ell}{k}\right) / 2^{\ell}$, *i.e.* the probability of a uniformly random string $S \xleftarrow{\$} \{0,1\}^{\ell}$ having Hamming distance at most t from a given $S_A \in \{0,1\}^{\ell}$.

$$\begin{aligned}
&\Pr[\mathsf{Game}^{\mathsf{C}^{\mathsf{B}}}_{\mathsf{NKD}}(1^\lambda) \not\Rightarrow 0] = \Pr[\mathsf{Game}\,3^{\mathsf{C}^{\mathsf{B}}}(1^\lambda) \not\Rightarrow 0] \\
&= \Pr[F \vee (\neg F \wedge \mathsf{C}^{\mathsf{B}}(1^\lambda) \Rightarrow b)] = \Pr[F] + \Pr[\neg F \wedge \mathsf{C}^{\mathsf{B}}(1^\lambda) \Rightarrow b] && (52)\\
&\geq \Pr[\neg F] \Pr[\mathsf{C}^{\mathsf{B}}(1^\lambda) \Rightarrow b \,|\, \neg F] && (53)\\
&= (1-\epsilon)\left(\Pr[\mathsf{C}^{\mathsf{B}}(1^\lambda) \Rightarrow b = 0 \,|\, \neg F] + \Pr[\mathsf{C}^{\mathsf{B}}(1^\lambda) \Rightarrow b = 1 \,|\, \neg F]\right) && (54)\\
&= \frac{1-\epsilon}{2}\left(\Pr[\mathsf{C}^{\mathsf{B}}(1^\lambda) \Rightarrow 0 \,|\, b = 0 \wedge \neg F] + \Pr[\mathsf{C}^{\mathsf{B}}(1^\lambda) \Rightarrow 1 \,|\, b = 1 \wedge \neg F]\right) && (55)\\
&= \tfrac{1-\epsilon}{2}\left(\Pr[\mathsf{B}(1^\lambda) \Rightarrow 0 \wedge \textsc{hw}(S \oplus S_A) > t \,|\, b = 0 \wedge \neg F] + \Pr[\mathsf{B}(1^\lambda) \Rightarrow 1 \,|\, b = 1 \wedge \neg F]\right) && (56)\\
&\geq \tfrac{1-\epsilon}{2}\left(\Pr[\mathsf{B}(1^\lambda) \Rightarrow 0 \,|\, b = 0 \wedge \neg F] \cdot (1-\phi) + \Pr[\mathsf{B}(1^\lambda) \Rightarrow 1 \,|\, b = 1 \wedge \neg F]\right) && (57)\\
&\geq \tfrac{1-\epsilon}{2}\left(\Pr[\mathsf{B}(1^\lambda) \Rightarrow 0 \,|\, b = 0 \wedge \neg F] + \Pr[\mathsf{B}(1^\lambda) \Rightarrow 1 \,|\, b = 1 \wedge \neg F]\right) \cdot (1-\phi) && (58)\\
&= (1-\phi) \cdot \Pr[\neg F] \cdot \Pr[\mathsf{B}(1^\lambda) \Rightarrow b] = (1 - \epsilon - \phi + \epsilon\phi) \cdot \Pr[\mathsf{Game}\,2^{\mathsf{B}}(1^\lambda) \Rightarrow 1] && (59)
\end{aligned}$$

Now describe $\mathsf{Adv}^{\mathsf{NKD}}_{\Pi}(\mathsf{C}) = \left|\Pr[\mathsf{NKD}^{\mathsf{C}^{\mathsf{B}}}_{\Pi}(1^\lambda) \not\Rightarrow 0] - \frac{1+\epsilon}{2}\right|$ in terms of $\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathcal{K}}(\mathsf{A})$. Then we get:

$$\begin{aligned}
&\mathsf{Adv}^{\mathsf{NKD}}_{\Pi}(\mathsf{C}) = \Pr[\mathsf{Game}\,3^{\mathsf{C}}(1^\lambda) \not\Rightarrow 0] - \frac{1+\epsilon}{2} && (60)\\
&\geq (1 - \epsilon - \phi + \epsilon\phi) \cdot \Pr[\mathsf{Game}\,2^{\mathsf{B}}(1^\lambda) \Rightarrow 1] - \frac{1+\epsilon}{2} && (61)\\
&\geq (1 - \epsilon - \phi + \epsilon\phi) \cdot \Bigg(\Pr[\mathsf{Game}\,1^{\mathsf{A}}(1^\lambda) \Rightarrow 1] - 2\hat{Q}_{\mathsf{H}_3}\Big(\sqrt{2(\hat{Q}_{\mathsf{H}_1}+1)\sqrt{2^{-\lambda}}} \\
&\quad + \sqrt{2(\hat{Q}_{\mathsf{H}_2}+1)\sqrt{2^{-\lambda}}} + 2\sqrt{\mathsf{Adv}^{\mathsf{NKD}}_{\Pi}}\Big)\Bigg) - \frac{1+\epsilon}{2}\ . && (62)
\end{aligned}$$

Isolate the term $\Pr[\mathsf{Game}\,1^{\mathsf{A}}(1^\lambda) \Rightarrow 1] = \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathcal{K}}(\mathsf{A})$ and use $\mathsf{Adv}^{\mathsf{NKD}}_{\Pi}(\mathsf{C}) \leq \mathsf{Adv}^{\mathsf{NKD}}_{\Pi}$. This yields the theorem statement:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathcal{K}}(\mathsf{A}) \leq{}& \frac{\mathsf{Adv}^{\mathsf{NKD}}_{\Pi} + \frac{1+\epsilon}{2}}{(1-\epsilon)(1-\phi)} - \frac{1}{2} + 2\hat{Q}_{\mathsf{H}_3}\Big(\sqrt{2(\hat{Q}_{\mathsf{H}_1}+1)\sqrt{2^{-\lambda}}} && (63)\\
&+ \sqrt{2(\hat{Q}_{\mathsf{H}_2}+1)\sqrt{2^{-\lambda}}} + 2\sqrt{\mathsf{Adv}^{\mathsf{NKD}}_{\Pi}}\Big)\ . && \square
\end{aligned}$$

6 Conclusion

This paper introduces the noisy key agreement (NKA) protocol as a standalone concept, and an appropriate security definition in the form of the NKD game. Furthermore, we present a transformation turning an NKA protocol into a key encapsulation mechanism (KEM) secure in the quantum random oracle model. The security proof relies on modeling the derandomization function H_3 as a variable output length random oracle. Along with new techniques for refined reasoning about the queries made by a quantum adversary and using the NKA protocol as a starting point, the resulting security bound relates the IND-CCA advantage of the assumed adversary to the insecurity of the NKD game.

The bound's reliance on the error probabiliy ϵ is to be expected because the occurrence of a protocol failure is equated to a complete loss of security. However, there is also a term involving ϕ, the probability of a uniformly random bitstring being less than t bits apart from a given one. The presence of this parameter is an artifact of the NKD formalism as $(1-\epsilon)(1-\phi)/2$ upper bounds any adversary's advantage in that game. In practice, both ϵ and ϕ should be made negligible in the security parameter.

Provided that this constraint is satisfied, our bound is much tighter than the those of Targhi-Unruh and Hofheinz *et al.* [61,35]. In particular, the term $\mathsf{Adv}^{\mathsf{NKD}}_{\Pi}$, which captures the insecurity of the underlying primitive, is degraded only by a square root, similar to the bound of Jiang *et al.* [39]. In contrast, the insecurity of the underlying primitive degrades with a quartic root in Targhi-Unruh and Hofheinz *et al.* All roots are the result either of the One-Way to Hiding Lemma or else of the One-Wayness game.

With respect to the concrete security of the Ramstake proposal, a couple of remarks are in order. First, the security bound explicitly features the error probability ϵ which in the case of Ramstake is rather high — roughly 2^{-64} for a security level of 128 bits against quantum computers. The bound therefore establishes less security than the claimed 128 bits. Nevertheless, when conditioning for the absence of decapsulation failures, the bottleneck becomes preimage search in a random function, and after that the NKD advantage. Moreover, it is by no means clear how much and even whether security is lost in the event of a decapsulation failure, although answering this question is a task for cryptanalysis rather than provable security.

Second, length of the hashes and seed is twice the claimed security level, in accordance with a speedup due to Grover's algorithm. However, the security degradation in the present bound resulting from these hash functions is a *fourth root*, much better than Grover's algorithm from the attacker's point of view. It remains an open question to determine whether this fourth root degradation is tight, *i.e.*, whether it can be matched by an attack. We note that Hülsing *et al.* [37] have a root-free insecurity function for preimage search applying specifically in the context of compressing hash functions. While their result does not apply in the present context, it is an uplifting indication that maybe the fourth root degradation is not a necessary quality of a security bound.

Acknowledgements

The authors would like to thank Aysajan Abidin, Chris Peikert, Mike Hamburg, Keita Xagawa, and others for useful comments and feedback, missing references, and pointing out flaws in proofs.

References

1. Aggarwal, D., Joux, A., Prakash, A., Santha, M.: A New Public-Key Cryptosystem via Mersenne Numbers. IACR Cryptology ePrint Archive 2017, 481, version 20170530:072202
2. Aguilar, C., Gaborit, P., Lacharme, P., Schrek, J., Zémor, G.: Noisy Diffie-Hellman Protocols (2010), `https://pqc2010.cased.de/rr/03.pdf`, PQCrypto 2010 (recent results session)
3. Aguilar, C., Gaborit, P., Lacharme, P., Schrek, J., Zémor, G.: Noisy Diffie-Hellman protocols or code-based key exchanged and encryption without masking (2010), `https://rump2010.cr.yp.to/fae8cd8265978675893352329786cea2.pdf`, CRYPTO 2010 (rump session)
4. Albrecht, M.R., Orsini, E., Paterson, K.G., Peer, G., Smart, N.P.: Tightly Secure Ring-LWE Based Key Encapsulation with Short Ciphertexts. In: Foley, S.N., Gollmann, D., Snekkenes, E. (eds.) ESORICS 2017, Part I. LNCS, vol. 10492, pp. 29–46. Springer (2017)
5. Alekhnovich, M.: More on average case vs approximation complexity. In: FOCS 2003. pp. 298–307. IEEE Computer Society (2003)
6. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: NewHope without reconciliation. IACR Cryptology ePrint Archive 2016, 1157
7. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum Key Exchange - A New Hope. In: Holz, T., Savage, S. (eds.) USENIX 2016. pp. 327–343. USENIX Association (2016)
8. Ambainis, A., Rosmanis, A., Unruh, D.: Quantum Attacks on Classical Proof Systems: The Hardness of Quantum Rewinding. In: IEEE FOCS 2014. pp. 474–483. IEEE Computer Society (2014)
9. Anada, H., Arita, S.: Identification Schemes from Key Encapsulation Mechanisms. IEICE Transactions 95-A(7), 1136–1155 (2012)
10. Barreto, P.S.L.M., Gueron, S., Gueneysu, T., Misoczki, R., Persichetti, E., Sendrier, N., Tillich, J.: CAKE: Code-based Algorithm for Key Encapsulation. IACR Cryptology ePrint Archive 2017, 757
11. Bellare, M., Hofheinz, D., Kiltz, E.: Subtleties in the Definition of IND-CCA: When and How Should Challenge Decryption Be Disallowed? J. Cryptology 28(1), 29–48 (2015)
12. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: CRYPTO '93. LNCS, vol. 773, pp. 232–249. Springer (1993)
13. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS '93. pp. 62–73. ACM (1993)
14. Bernstein, D.J., Chou, T., Schwabe, P.: McBits: Fast Constant-Time Code-Based Cryptography. In: Bertoni, G., Coron, J. (eds.) CHES 2013. LNCS, vol. 8086, pp. 250–272. Springer (2013)

15. Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random Oracles in a Quantum World. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 41–69. Springer (2011)
16. Bos, J.W., Costello, C., Ducas, L., Mironov, I., Naehrig, M., Nikolaenko, V., Raghunathan, A., Stebila, D.: Frodo: Take off the Ring! Practical, Quantum-Secure Key Exchange from LWE. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 1006–1018. ACM (2016)
17. Bos, J.W., Costello, C., Naehrig, M., Stebila, D.: Post-Quantum Key Exchange for the TLS Protocol from the Ring Learning with Errors Problem. In: IEEE S&P 2015. pp. 553–570. IEEE Computer Society (2015)
18. Bos, J.W., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Stehlé, D.: CRYSTALS - Kyber: a CCA-secure module-lattice-based KEM. IACR Cryptology ePrint Archive 2017, 634
19. Boyd, C., Cliff, Y., Nieto, J.M.G., Paterson, K.G.: One-round key exchange in the standard model. IJACT 1(3), 181–199 (2009)
20. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer (2001)
21. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: an efficient post-quantum commutative group action. IACR Cryptology ePrint Archive 2018, 383 (2018), `https://eprint.iacr.org/2018/383`
22. Cramer, R., Shoup, V.: A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In: CRYPTO '98
23. Cramer, R., Shoup, V.: Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. SIAM J. Comput. 33(1), 167–226 (2003)
24. Cremers, C.J.F., Feltz, M.: Beyond eCK: Perfect Forward Secrecy under Actor Compromise and Ephemeral-Key Reveal. Des. Codes Cryptography 74(1), 183–218 (2015)
25. Dagdelen, Ö., Fischlin, M., Gagliardoni, T.: The Fiat-Shamir Transformation in a Quantum World. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 62–81. Springer (2013)
26. Deneuville, J., Gaborit, P., Zémor, G.: Ouroboros: A Simple, Secure and Efficient Key Exchange Protocol Based on Coding Theory. In: Lange, T., Takagi, T. (eds.) PQCrypto 2017. LNCS, vol. 10346, pp. 18–34. Springer (2017)
27. Dent, A.W.: A Designer's Guide to KEMs. In: Paterson, K.G. (ed.) IMA 9th Conf. Cryptography and Coding. LNCS, vol. 2898, pp. 133–151. Springer (2003)
28. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Information Theory 22(6), 644–654 (1976)
29. Ding, J., Lie, X., Lin, X.: A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem. IACR Cryptology ePrint Archive 2012, 688 (2012)
30. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO '86. LNCS, vol. 263, pp. 186–194. Springer (1986)
31. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly secure authenticated key exchange from factoring, codes, and lattices. Des. Codes Cryptography 76(3), 469–504 (2015)
32. Fujisaki, E., Okamoto, T.: How to Enhance the Security of Public-Key Encryption at Minimum Cost. In: Imai, H., Zheng, Y. (eds.) PKC '99. LNCS, vol. 1560, pp. 53–68. Springer (1999)

33. Grover, L.K.: A Fast Quantum Mechanical Algorithm for Database Search. In: Miller, G.L. (ed.) ACM STOC 1996. pp. 212–219. ACM (1996)
34. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A Ring-Based Public Key Cryptosystem. In: Buhler, J. (ed.) ANTS-III, 1998. LNCS, vol. 1423, pp. 267–288. Springer (1998)
35. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A Modular Analysis of the Fujisaki-Okamoto Transformation. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 341–371. Springer (2017)
36. Hülsing, A., Rijneveld, J., Schanck, J.M., Schwabe, P.: High-Speed Key Encapsulation from NTRU. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 232–252. Springer (2017)
37. Hülsing, A., Rijneveld, J., Song, F.: Mitigating Multi-target Attacks in Hash-Based Signatures. In: Cheng, C., Chung, K., Persiano, G., Yang, B. (eds.) PKC 2016, Part I. LNCS, vol. 9614, pp. 387–416. Springer (2016)
38. Jao, D., Feo, L.D.: Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies. In: Yang, B. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 19–34. Springer (2011)
39. Jiang, H., Zhang, Z., Chen, L., Wang, H., Ma, Z.: Post-quantum IND-CCA-secure KEM without additional hash. IACR Cryptology ePrint Archive 2017, 1096 (2017), http://eprint.iacr.org/2017/1096
40. Jin, Z., Zhao, Y.: Optimal Key Consensus in Presence of Noise. CoRR abs/1611.06150 (2016)
41. Krawczyk, H.: SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE-Protocols. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 400–425. Springer (2003)
42. LaMacchia, B.A., Lauter, K.E., Mityagin, A.: Stronger Security of Authenticated Key Exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer (2007)
43. Lindner, R., Peikert, C.: Better key sizes (and attacks) for lwe-based encryption. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 319–339. Springer (2011)
44. Lyubashevsky, V., Peikert, C., Regev, O.: On Ideal Lattices and Learning with Errors over Rings. J. ACM 60(6), 43:1–43:35 (2013)
45. Matsumoto, T., Imai, H.: Public Quadratic Polynominal-Tuples for Efficient Signature-Verification and Message-Encryption. In: Günther, C.G. (ed.) EUROCRYPT '88. LNCS, vol. 330, pp. 419–453. Springer (1988)
46. McEliece, R.J.: A public-key cryptosystem based on algebraic coding theory. DNS Progress Report 4244, 114–116 (1978)
47. Melchor, C.A., Blazy, O., Deneuville, J., Gaborit, P., Zémor, G.: Efficient Encryption from Random Quasi-Cyclic Codes. CoRR abs/1612.05572 (2016)
48. National Institute for Standards and Technology (NIST): FIPS PUB 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions (2015)
49. National Institute for Standards and Technology (NIST): Post-quantum crypto standardization (2018), http://csrc.nist.gov/groups/ST/post-quantum-crypto/
50. National Institute for Standards and Technology (NIST): Submission to the NIST call for PQC proposals. (2018), https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions
51. Niederreiter, H.: Knapsack-type cryptosystems and algebraic coding theory. Problems of Control and Information Theory. Problemy Upravlenija i Teorii Informacii. 15, 159–166 (1986)

52. Nielsen, M.A., Chuang, I.L.: Quantum computation and quantum information. Cambridge university press (2010)
53. Peikert, C.: Some recent progress in lattice-based cryptography. In: Reingold, O. (ed.) TCC 2009. Lecture Notes in Computer Science, vol. 5444. Springer (2009), invited talk.
54. Peikert, C.: Lattice Cryptography for the Internet. In: Mosca, M. (ed.) PQCrypto 2014. LNCS, vol. 8772, pp. 197–219. Springer (2014)
55. Porras, J., Baena, J., Ding, J.: Zhfe, a new multivariate public key encryption scheme. In: Mosca, M. (ed.) PQCrypto 2014. Lecture Notes in Computer Science, vol. 8772, pp. 229–245. Springer (2014)
56. Rackoff, C., Simon, D.R.: Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In: Feigenbaum, J. (ed.) CRYPTO '91. LNCS, vol. 576, pp. 433–444. Springer (1991)
57. Shor, P.W.: Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In: FOCS 35. pp. 124–134. IEEE Computer Society (1994)
58. Song, F.: A Note on Quantum Security for Post-Quantum Cryptography. In: Mosca, M. (ed.) PQCrypto 2014. LNCS, vol. 8772, pp. 246–265. Springer (2014)
59. Szepieniec, A., Ding, J., Preneel, B.: Extension Field Cancellation: A New Central Trapdoor for Multivariate Quadratic Systems. In: Takagi, T. (ed.) PQCrypto 2016. LNCS, vol. 9606, pp. 182–196. Springer (2016)
60. Tao, C., Diene, A., Tang, S., Ding, J.: Simple Matrix Scheme for Encryption. In: PQCrypto 2013
61. Targhi, E.E., Unruh, D.: Post-Quantum Security of the Fujisaki-Okamoto and OAEP Transforms. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 192–216 (2016)
62. Tolhuizen, L., Rietman, R., García-Morchón, Ó.: Improved key-reconciliation method. IACR Cryptology ePrint Archive 2017, 295
63. Unruh, D.: Non-Interactive Zero-Knowledge Proofs in the Quantum Random Oracle Model. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 755–784. Springer (2015)
64. Unruh, D.: Revocable quantum timed-release encryption. J. ACM 62(6), 49:1–49:76 (2015), http://doi.acm.org/10.1145/2817206
65. Unruh, D.: Computationally Binding Quantum Commitments. In: Fischlin, M., Coron, J. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 497–527. Springer (2016)
66. Yoneyama, K.: Compact Authenticated Key Exchange from Bounded CCA-Secure KEM. IEICE Transactions 98-A(1), 132–143 (2015)
67. Zhandry, M.: Secure identity-based encryption in the quantum random oracle model. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 758–775. Springer (2012)

A Concrete Instantiations of NKA Protocols

We now consider several concrete instantiations of noisy key agreement that are used in the literature to generate key encapsulation mechanisms or public key encryption schemes. In all cases, the participants of the protocol converge to mathematical objects whose distance is small in some sense. We make abstraction of this notion of smallness and represent the mathematical objects as bitstrings (denoted by $\llcorner\cdot\lrcorner$) at which point the Hamming weight metric can be used.

It is worth emphasizing that the concrete problems we identify must be assumed to be hard, even in the context of quantum computers, in order for the protocol and KEM or PKE to be secure. Nevertheless, the NKD Assumption is the only requirement; the other problems are hard on average if the NKD Assumption is true.

NewHope [7]. NewHope defines a ring $\mathcal{R}_q \cong \mathbb{Z}[X]/\langle q, X^n + 1\rangle$ and a centered binomial distrubution Ψ_{16}^n over $\mathcal{R}_q$. Elements that are sampled according to Ψ_{16}^n are considered small. The protocol functionalities and noisy key views are as follows.

Init:	generate $\mathbf{a} \in \mathcal{R}_q$ from *seed*
AContr:	sample $\mathbf{s}, \mathbf{e} \sim \Psi_{16}^n$ and transmit $\mathbf{b} = \mathbf{as} + \mathbf{e}$
BContr:	sample $\mathbf{s}', \mathbf{e}' \sim \Psi_{16}^n$ and transmit $\mathbf{u} = \mathbf{as}' + \mathbf{e}'$
AConv:	compute $\mathbf{v} = \mathbf{us}$
BConv:	compute $\mathbf{v}' = \mathbf{bs}'$
S_A:	$\llcorner\mathbf{v}\lrcorner$
S_B:	$\llcorner\mathbf{v}'\lrcorner$

This description gives rise to the following hard problems. The state recovery problems are instances of Ring-LWE.

A State Recovery (ASR).
Input: $\mathbf{a}, \mathbf{b} \in \mathcal{R}_q$ s.t. $\mathbf{b} = \mathbf{as} + \mathbf{e}$ for some $\mathbf{e}, \mathbf{s} \sim \Psi_{16}^n$
Task: find $\mathbf{s}, \mathbf{e} \sim \Psi_{16}^n$ s.t. $\mathbf{b} = \mathbf{as} + \mathbf{e}$

B State Recovery (BSR).
Input: $\mathbf{a}, \mathbf{u} \in \mathcal{R}_q$ s.t. $\mathbf{u} = \mathbf{as}' + \mathbf{e}'$ for some $\mathbf{e}', \mathbf{s}' \sim \Psi_{16}^n$
Task: find $\mathbf{s}, \mathbf{e} \sim \Psi_{16}^n$ s.t. $\mathbf{u} = \mathbf{as}' + \mathbf{e}'$

Noisy Key Search (NKS).
Input: $\mathbf{a}, \mathbf{b}, \mathbf{u} \in \mathcal{R}_q$ such that $\mathbf{b} = \mathbf{as} + \mathbf{e}$ and $\mathbf{u} = \mathbf{as}' + \mathbf{e}'$ for some $\mathbf{s}, \mathbf{s}', \mathbf{e}, \mathbf{e}' \sim \Psi_{16}^n$
Task: find $S \in \{0,1\}^\ell$ such that $\textsc{hw}(S \oplus \llcorner\mathbf{v}\lrcorner) \leq t$ and $\textsc{hw}(S \oplus \llcorner\mathbf{v}'\lrcorner) \leq t$, where $\mathbf{v} = \mathbf{us}$ and $\mathbf{v}' = \mathbf{bs}'$.

Noisy Key Distinguishing (NKD).
Input: $\mathbf{a}, \mathbf{b}, \mathbf{u} \in \mathcal{R}_q$ and $S \in \{0,1\}^\ell$ such that $\mathbf{b} = \mathbf{as} + \mathbf{e}$ and $\mathbf{u} = \mathbf{as}' + \mathbf{e}'$ for some $\mathbf{s}, \mathbf{s}', \mathbf{e}, \mathbf{e}' \sim \Psi_{16}^n$
Task: output 1 if $\textsc{hw}(S \oplus \llcorner\mathbf{v}\lrcorner) \leq t$ and $\textsc{hw}(S \oplus \llcorner\mathbf{v}'\lrcorner) \leq t$, where $\mathbf{v} = \mathbf{us}$ and $\mathbf{v}' = \mathbf{bs}'$; and 0 otherwise.

Ramstake [50]. Ramstake operates on integers modulo a large Mersenne prime p, the set of which we denote by $\mathbb{Z}_p$. Smallness is associated with having a bit expansion of low Hamming weight. We denote this set of sparse integers by $\mathcal{S}$. The functionalities and noisy key views are as follows.

Init:	sample $A \in \mathbb{Z}_p$
AContr:	sample $b, c \xleftarrow{\$} \mathcal{S}$ and transmit $D = Ab + c \bmod p$
BContr:	sample $b', c' \xleftarrow{\$} \mathcal{S}$ and transmit $D' = Ab' + c' \bmod p$
AConv:	compute $E = D'a \bmod p$
BConv:	compute $E' = Da' \bmod p$
S_A:	$\llcorner E \lrcorner$
S_B:	$\llcorner E' \lrcorner$

The problems of recovering either participant's state is in fact an affine variant of the low-Hamming weight ratio problem introduced by Aggarwal *et al.* [1]. Paraphrased but without loss of generality, this problem asks to find low-Hamming-weight integers f and g such that the given integer H satisfies $f \times (-H) + g = 0 \bmod p$.

A State Recovery (ASR).
Input: $A, D \in \mathbb{Z}_p$ s.t. $D = Ab + c$ for some $b, c \in \mathcal{S}$
Task: find $b, c \in \mathcal{S}$ s.t. $D = Ab + c$

B State Recovery (BSR).
Input: $A, D' \in \mathbb{Z}_p$ s.t. $D' = Ab' + c'$ for some $b', c' \in \mathcal{S}$
Task: find $b', c' \in \mathcal{S}$ s.t. $D' = Ab' + c'$

Noisy Key Search (NKS).
Input: $A, D, D' \in \mathbb{Z}_p$ such that $D = Ab + c$ and $D' = Ab' + c'$ for some $b, c, b', c' \in \mathcal{S}$
Task: find $S \in \{0,1\}^\ell$ such that $\textsc{hw}(S \oplus \llcorner E \lrcorner) \leq t$ and $\textsc{hw}(S \oplus \llcorner E' \lrcorner) \leq t$, where $E = D'a$ and $E' = Da'$.

Noisy Key Distinguishing (NKD).
Input: $A, D, D' \in \mathbb{Z}_p, S \in \{0,1\}^\ell$ such that $D = Ab + c$ and $D' = Ab' + c'$ for some $b, c, b', c' \in \mathcal{S}$
Task: output 1 if $\textsc{hw}(S \oplus \llcorner E \lrcorner) \leq t$ and $\textsc{hw}(S \oplus \llcorner E' \lrcorner) \leq t$, where $E = D'a$ and $E' = Da'$; and 0 otherwise.

Ouroboros [26]. Ouroboros uses the ring $\mathcal{R} = \mathbb{F}_2[X]/\langle X^n - 1 \rangle$, in which elements are considered small if their Hamming weight is less than a given bound. Let $\mathcal{S}_w^n \subset \mathcal{R}$ denote the subset of ring elements whose Hamming weight is w. The functionalities and noisy key views are as follows.

Init:	generate $\mathbf{h} \in \mathcal{R}$ from *seed*
AContr:	sample $\mathbf{x}, \mathbf{y} \xleftarrow{\$} \mathcal{S}_w^n$ and transmit $\mathbf{s} = \mathbf{xh} + \mathbf{y}$
BContr:	sample $\mathbf{r}_1, \mathbf{r}_2 \xleftarrow{\$} \mathcal{S}_w^n$ and transmit $\mathbf{s}_r = \mathbf{r}_1 + \mathbf{hr}_2$
AConv:	compute $S_A = \mathbf{ys}_r$
BConv:	compute $S_B = \mathbf{sr}_2$
S_A:	$\llcorner S_A \lrcorner$
S_B:	$\llcorner S_B \lrcorner$

While the values S_A and S_B are computed, both are instantly added to other values. Bob obtains $\mathbf{s}_\epsilon = S_B + \mathbf{e}_r + \boldsymbol{\epsilon}$ for specific values of $\mathbf{e}_r$ and $\boldsymbol{\epsilon}$, and transmits this value alongside $\mathbf{s}_r$. Alice obtains $\mathbf{e}_c = \mathbf{s}_\epsilon - S_A$, which is a noisy codeword from which the specialized decoder can recover $\boldsymbol{\epsilon}$. Ouroboros thus uses the transmission-based approach, and makes clever use of the decoder provided by the algebraic structure on which the noisy key agreement protocol is based.

A State Recovery (ASR).
Input: $\mathbf{h}, \mathbf{s} \in \mathcal{R}$ s.t. $\mathbf{s} = \mathbf{xh} + \mathbf{y}$ for some $\mathbf{x}, \mathbf{y} \in \mathcal{S}_w^n$
Task: find $\mathbf{x}, \mathbf{y} \in \mathcal{S}_2^n$ s.t. $\mathbf{s} = \mathbf{hx} + \mathbf{y}$

B State Recovery (BSR).
Input: $\mathbf{h}, \mathbf{s}_r \in \mathcal{R}$ s.t. $\mathbf{s}_r = \mathbf{r}_2\mathbf{h} + \mathbf{r}_2$ for some $\mathbf{r}_1, \mathbf{r}_2 \in \mathcal{S}_w^n$
Task: find $\mathbf{r}_1, \mathbf{r}_2 \in \mathcal{S}_2^n$ s.t. $\mathbf{s}_2 = \mathbf{hr}_2 + \mathbf{r}_1$

Noisy Key Search (NKS).
Input: $\mathbf{h}, \mathbf{s}, \mathbf{s}_r \in \mathcal{R}$ such that $\mathbf{s} = \mathbf{xh} + \mathbf{y}$ and $\mathbf{s}_r = \mathbf{hr}_2 + \mathbf{r}_1$ for some $\mathbf{x}, \mathbf{y}, \mathbf{r}_1, \mathbf{r}_2 \in \mathcal{S}_w^n$
Task: find $S \in \{0,1\}^\ell$ such that $\text{HW}(S \oplus \llcorner S_A \lrcorner) \leq t$ and $\text{HW}(S \oplus \llcorner S_B \lrcorner) \leq t$, where $S_A = \mathbf{s}_r\mathbf{x}$ and $S_B = \mathbf{sr}_2$.

Noisy Key Distinguishing (NKD).
Input: $\mathbf{h}, \mathbf{s}, \mathbf{s}_r \in \mathcal{R}, S \in \{0,1\}^\ell$ such that $\mathbf{s} = \mathbf{xh} + \mathbf{y}$ and $\mathbf{s}_r = \mathbf{hr}_2 + \mathbf{r}_1$ for some $\mathbf{x}, \mathbf{y}, \mathbf{r}_1, \mathbf{r}_2 \in \mathcal{S}_w^n$
Task: output 1 if $\text{HW}(S \oplus \llcorner S_A \lrcorner) \leq t$ and $\text{HW}(S \oplus \llcorner S_B \lrcorner) \leq t$, where $S_A = \mathbf{s}_r\mathbf{x}$ and $S_B = \mathbf{sr}_2$; and 0 otherwise.

SIDH [38]. The supersingular isogeny Diffie-Hellman (SIDH) is the only noise-free key agreement protocol on this list, and as such achieves identical views on the session key for both parties. The protocol relies on the commutativity of random walks in an isogeny graph of supersingular elliptic curves. We use the following standard notation, denoting elliptic curves by E; k-order torsion subgroups by $E[k]$; isogenies by ψ, ϕ; base points by P, Q; j-invariant by $j(\cdot)$. Generally speaking, $\ell_A = 2$ and $\ell_B = 3$ and the exponents e_A and e_B are large, say on the order of several hundreds. $P_A, Q_A \in E[\ell_A^{e_A}]$ are elements of the $\ell_A^{e_A}$-order torsion subgroup of E, and vice versa for B. The protocol's functionalities and session key can be summarized as follows.

Init: select $E_0 \xleftarrow{\$} \mathbb{E}(\mathbb{F}_q)$; $P_A, Q_A \xleftarrow{\$} E_0[\ell_A^{e_A}]$; $P_B, Q_B \xleftarrow{\$} E_0[\ell_B^{e_B}]$

AContr: sample $m_A, n_A \xleftarrow{\$} \mathbb{Z}/{\ell_A}^{e_A}\mathbb{Z}$; compute $R_A = m_A P_A + n_A Q_A$;
find $\phi : \mathbb{E}(\mathbb{F}_q) \to \mathbb{E}(\mathbb{F}_q)$ such that $\ker \phi = \langle R_A \rangle$;
transmit $E_A = \phi(E_0), \phi(P_B), \phi(Q_B)$

BContr: sample $m_B, n_B \xleftarrow{\$} \mathbb{Z}/{\ell_B}^{e_B}\mathbb{Z}$; compute $R_B = m_B P_B + n_B Q_B$;
find $\psi : \mathbb{E}(\mathbb{F}_q) \to \mathbb{E}(\mathbb{F}_q)$ such that $\ker \psi = \langle R_B \rangle$;
transmit $E_B = \psi(E_0), \psi(P_A), \psi(Q_A)$

AConv: compute $R'_A = n_A \psi(P_A) + m_A \psi(Q_A) \in E_B$;
find $\phi' : \mathbb{E}(\mathbb{F}_q) \to \mathbb{E}(\mathbb{F}_q)$ such that $\ker \phi' = \langle R'_A \rangle$;
compute $E_{BA} = \phi'(E_B)$

BConv: compute $R'_B = n_B \phi(P_B) + m_B \phi(Q_B) \in E_A$;
find $\psi' : \mathbb{E}(\mathbb{F}_q) \to \mathbb{E}(\mathbb{F}_q)$ such that $\ker \psi' = \langle R'_B \rangle$;
compute $E_{AB} = \psi'(E_A)$

S_A: $\llcorner j(E_{BA}) \lrcorner$

S_B: $\llcorner j(E_{AB}) \lrcorner$

The original SIDH paper already explicitly considers the hard problems associated with the protocol. They are called the Computational Supersingular Isogeny (CSSI) problem for ASR or BSR; Supersingular Computational Diffie-Hellman (SSCDH) problem for NKS; and Supersingular Decisional Diffie-Hellman (SSDDH) problem for NKD. We adopt this nomenclature.

Computational Supersingular Isogeny Problem (CSSI).
Input: $E_0, E_A = \phi(E_0); P_B, Q_B, P_A, Q_A \in E_0; \phi(P_B), \phi(Q_B) \in E_A$ for some isogeny $\phi : E_0 \to E_A$ with $\ker \phi = \langle n_A P_A + m_A Q_A \rangle$
Task: find a generator for $\langle R \rangle = \langle n_A P_A + m_A P_A \rangle = \ker \phi$

Supersingular Isogeny Computational Diffie-Hellman (SSCDH) Problem.
Input: $E_0, E_A = \phi(E_0), E_B = \psi(E_0)$; $P_A, Q_A P_B, Q_B \in E_0$; $\phi(P_B), \phi(Q_B) \in E_A; \psi(P_A), \psi(Q_A) \in E_B$ for isogenies $\phi, \psi : E_0 \to E_A$ with $\ker \phi = \langle n_A P_A + m_A Q_A \rangle$ and $\ker \psi = \langle n_B P_B + m_B Q_B \rangle$
Task: find $j(E_{AB})$ where $E_{AB} \cong E_0/\langle n_A P_A + m_A Q_A + n_B P_B + m_B Q_B \rangle$.

Supersingular Isogeny Decisional Diffie-Hellman (SSDDH) Problem.
Input: $E_0, E_A = \phi(E_0), E_B = \psi(E_0)$; $P_A, Q_A P_B, Q_B \in E_0$; $\phi(P_B), \phi(Q_B) \in E_A; \psi(P_A), \psi(Q_A) \in E_B; j \in \mathbb{F}_q$ for isogenies $\phi, \psi : E_0 \to E_A$ with $\ker \phi = \langle n_A P_A + m_A Q_A \rangle$ and $\ker \psi = \langle n_B P_B + m_B Q_B \rangle$
Task: output 1 if $j = j(E_0/\langle n_A P_A + m_A Q_A + n_B P_B + m_B Q_B \rangle)$; and 0 otherwise.

B Noisy Key Security

In the previous we have defined, with some justification, the security of NKA protocols in terms of the NKD game. Here we extend this justification by considering the most general possible security definition, *i.e.*, an adaptation of the Canetti-Krawczyk session key security (SK-security) notion in the authenticated links model [20], which we call *noisy key security (NK-security)*. It turns out that NK-security is equivalent to the hardness of NKD, up to a polynomial factor.

B.1 NK-Security

Adapting SK-security to the noisy case presents two difficulties.

First, Alice and Bob do not agree on the same key but on two different views S_A and S_B which are close under the Hamming metric. The adversary is deemed successful if he can distinguish between a uniformly random key and one drawn at random from the intersection of radius-t spheres centered at S_A and S_B. This extension captures the special case of noise-free key agreement of the Canetti-Krawczyk model, in which this intersection collapses to a single point $S_A = S_B$.

Second, there is a small but nonzero probability of failure even when the adversary does not interfere and it is conceivable that approximating either Alice's view or Bob's view of the session key is easier in this case. To deal with this issue, the security game aborts when the adversary picks a failing game. This choice is the same for the NKD game.

Like Canetti-Krawczyk's definition, ours considers an adversary A and any number of parties P_i each pair of which can run any number of *sessions*. The adversary can

- see, block, resend all messages passed between parties (but not modify them);
- schedule events, *i.e.*, instruct parties to start sessions or proceed with the next step;
- expire sessions, *i.e.*, instruct parties to forget the agreed-upon session key or associated state;
- *expose* sessions, either though
  - *session-state reveal*, which reveals a party's session state; or
  - *session-key query*, which reveals one party's view of the session key; or
  - *corruption*, in which case the adversary learns the entire working memory of a targeted party whose subsequent actions are all directed by the adversary.

The adversary chooses among all the unexposed sessions one *test session*, and if this test session is unsuccessful ($\textsc{hw}(S_A \oplus S_B) > t$) the game aborts. Otherwise the adversary receives a string S which is, depending on a coin flip $b \xleftarrow{\$} \{0,1\}$, either either drawn from the intersection of radius-t spheres centered at S_A and S_B, or uniformly at random from the set of all bit strings of the same length. The adversary outputs a bit $\hat{b}$ guessing at the distribution from which S was

drawn; he wins if he guesses correctly. The protocol is *noisy key secure* in the authenticated links model if no polynomial-time quantum adversary has more than a negligible distinguishing advantage. This notion is captured in words by Definition 4. Pseudocode for the oracles' behavior and the game mechanics is given in Appendix B.2.

Definition 4 (noisy key security). *Let* $\Pi = (\mathsf{Init}, \mathsf{AContr}, \mathsf{BContr}, \mathsf{AConv}, \mathsf{BConv})$ *be a noisy key agreement protocol between parties* P_A *and* P_B*, with correctness error* ϵ*. The game* NK *defines an adversary* $\mathsf{A}^{\cdots} = (\mathsf{A}_1^{\cdots}, \mathsf{A}_2^{\cdots})$ *with oracle access to the following functions:*

- $\mathsf{start}(P_A, P_B)$ *instructs parties* P_A *and* P_B *to start a new session with a fresh session id;*
- $\mathsf{deliver}(\mathit{receiver}, \mathit{sender}, \mathit{session\_id}, \mathit{contribution})$ *delivers the unaltered contribution message from receiver to sender if both are involved in session id session_id;*
- $\mathsf{contribute}(\mathit{party}, \mathit{session\_id})$ *instructs participant party to generate a contribution message for session session_id;*
- $\mathsf{converge}(\mathit{party}, \mathit{session\_id}, \mathit{contribution})$ *instructs participant party to converge, and thus obtain their view of the shared noisy session key;*
- $\mathsf{expire}(\mathit{party}, \mathit{session\_id})$ *instructs participant party to consider session session_id expired, that is to say inactive for all intents and purposes;*
- $\mathsf{reveal\_state}(\mathit{party}, \mathit{session})$ *reveals the secret state of participant party for session session_id, but as a result the session becomes exposed;*
- $\mathsf{query\_key}(\mathit{party}, \mathit{session\_id})$ *reveals party's view of the shared noisy session key from session session_id, but as a result the session becomes exposed;*
- $\mathsf{corrupt}(\mathit{party}, \mathit{code})$ *instructs participant party to execute code with access to party's state and with capability to send authentic-looking messages on behalf of party, but as a result all of party's sessions become exposed.*

The NK *game proceeds in two phases: in phase 1, the adversary* $\mathsf{A}_1^{\cdots}$ *runs with access to all the above oracles for a polynomially bounded number of time steps and as a result outputs a secret state and a test session test_session_id. If session test_session_id fails* ($\textsc{hw}(S_A \oplus S_B) > t$) *or if it has been exposed (through an invocation of* reveal_state, query_key, *or* corrupt*) then the game aborts and outputs* $\perp$*. Phase 2 starts when the challenger flips a coin* b *and if* $b = 0$ *he sets* $S \xleftarrow{\$} \{0,1\}^\ell$ *but if* $b = 1$ *then* $S \xleftarrow{\$} \{x \in \{0,1\}^\ell \mid \textsc{hw}(x \oplus S_A) \leq t \wedge \textsc{hw}(x \oplus S_B) \leq t\}$ *where* S_A *and* S_B *are the views of the shared noisy session key of parties* P_A *and* P_B *associated with session test_session_id. Then* $\mathsf{A}_2^{\cdots}$ *is run with access to all oracles on input* (state, S) *for another polynomially bounded number of steps, after which he outputs a guess* $\hat{b}$*. The game outputs 1 if* $b = \hat{b}$ *and 0 otherwise.*

Then the noisy key agreement protocol is noisy key secure *(NK-secure) in the* authenticated links model *if for all polynomial time quantum adversaries* $\mathsf{A}^{\cdots}$ *who starts* k *sessions and corrupts* r *of them, their advantage* $\mathsf{Adv}_{\Pi}^{\mathsf{NK}}(\mathsf{A}^{\cdots})$ *is negligible:*

$$\mathsf{Adv}_{\Pi}^{\mathsf{NK}}(\mathsf{A}^{\cdots}) \triangleq \left| \Pr[\mathsf{Game}_{\mathsf{NK}}^{\mathsf{A}}(1^\lambda) \not\Rightarrow 0] - \frac{1 + \epsilon^{\frac{k-r}{k}} + \frac{r}{k}}{2} \right| \leq \mathsf{negl}(\lambda) \ . \tag{64}$$

This expression for the adversary's advantage is rather complex but certainly valid. The adversary who corrupts every session he starts in order to engineer game abortions, has advantage zero. The same is true for the adversary who does not corrupt any session but flips a coin and guesses accordingly. The adversary's advantage remains zero for any combination of these two extremes. Therefore, the expression captures the adversary's advantage over a naïve strategy.

We stress that a NKA protocol must consist of two independent messages, one in each direction, as formalized in the syntax. While the Canetti-Krawczyk security model does not impose any bounds on the number of messages exchanged or their scheduling, in the case of NKA this restriction on the number of passes is critical; if the parties involved are allowed more then they can agree on an exact key simply by transmitting auxiliary information to correct errors.

Game 9: $\mathsf{NK}^{\mathsf{A}\cdots}(1^\lambda)$

1. $party\_states \leftarrow [\varnothing$ **for all** parties$]$
2. $authentic\_messages, global\_sessions \leftarrow \mathsf{empty\_lists}$
3. $session\_counter \leftarrow 0$
4. $test\_session\_id, state \leftarrow \mathsf{A}_1^{\cdots}(1^\kappa)$
5. **if** $global\_sessions[test\_session\_id].exposed = \mathsf{True}$ **then:**
6. $\quad$ **return** $\perp$
7. $P_A \leftarrow global\_sessions[test\_session\_id].A$
8. $P_B \leftarrow global\_sessions[test\_session\_id].B$
9. $S_A \leftarrow party\_states[P_A].sessions[test\_session\_id].S$
10. $S_B \leftarrow party\_states[P_B].sessions[test\_session\_id].S$
11. **if** $\mathrm{HW}(S_A \oplus S_B) > t$ **then:**
12. $\quad$ **return** $\perp$
13. $b \xleftarrow{\$} \{0,1\}$
14. **if** $b = 1$ **then:**
15. $\quad S \xleftarrow{\$} \{x \in \{0,1\}^\ell \mid \mathrm{HW}(x \oplus S_A) \wedge \mathrm{HW}(x \oplus S_B)\}$
16. **else:**
17. $\quad S \xleftarrow{\$} \{0,1\}^\ell$
18. $\hat{b} \leftarrow \mathsf{A}_2^{\cdots}(state, S)$
19. **return** $[\![b = \hat{b}]\!]$

B.2 Pseudocode for Oracle Behavior

Oracle 10: deliver($\cdot$)

```
1. define deliver(receiver, sender, session_id, contribution) as:
2.     if (receiver, sender, session_id, contribution) ∉ authentic_messages then:
3.         return ⊥
4.     if session_id ∉ party_states[receiver].sessions.keys() then:
5.         return ⊥
6.     if session_id ∉ party_states[sender].sessions.keys() then:
7.         return ⊥
8.     party_states[receiver].sessions[session_id].contribution = contribution
```

Oracle 11: start($\cdot$)

```
1. define start(P_A, P_B) as:
2.     global_sessions.append(global_session(
3.         A = P_A,
4.         B = P_B,
5.         exposed = False))
6.     iparams ← Π.Init(1^κ)
7.     party_states[P_A].sessions.append(session(
8.         key = session_counter,
9.         A = P_A,
10.        B = P_B,
11.        params = iparams,
12.        state = ∅,
13.        contribution = ∅,
14.        S = 0^ℓ))
15.    party_states[P_B].sessions.append(session(
16.        key = session_counter,
17.        A = P_A,
18.        B = P_B,
19.        params = iparams,
20.        state = ∅,
21.        contribution = ∅,
22.        S = 0^ℓ))
23.    session_counter ← session_counter + 1
```

Oracle 12: contribute(·)

```
1. define contribute(party, session_id) as:
2.     if session_id ∉ party_states[party].sessions.keys() then:
3.         return ⊥
4.     session ← party_states[party].sessions[session_id]
5.     if party = session.A then:
6.         session.state, session.contribution ← Π.AContr(session.params)
7.     else:
8.         session.state, session.contribution ← Π.BContr(session.params)
9.     party_states[party].sessions[session_id] ← session
10.    msg ← message(
11.        sender = party,
12.        receiver = {session.A, session.B}\party,
13.        session_id = session_id,
14.        contribution = session.contribution)
15.    authentic_messages.append(msg)
16.    return msg    ▷ allow adversary to block
```

Oracle 13: converge(·)

```
1. define converge(party, session_id, contribution) as:
2.     if session_id ∉ party_states[party].sessions.keys() then:
3.         return ⊥
4.     session ← party_states[party].sessions[session_id]
5.     other ← {session.A, session.B}\party
6.     if (other, party, session_id, contribution) ∉ authentic_messages then:
7.         return ⊥
8.     if party = session.A then:
9.         session.S ← Π.AConv(session.state, contribution)
10.    else:
11.        session.S ← Π.BConv(session.state, contribution)
12.    session.state = ∅
13.    party_states[party].sessions[session_id] ← session
```

Oracle 14: expire(·)

```
1. define expire(party, session_id) as:
2.     if session_id ∉ party_states[party].sessions.keys then:
3.         return ⊥
4.     party_states[party].sessions[session_id].S = ∅
```

Oracle 15: reveal_state(·)

```
1. define reveal_state(party, session_id) as:
2.     if session_id ∉ party_states[party].sessions.keys then:
3.         return ⊥
4.     global_sessions[session_id].exposed ← True
5.     return party_states[party].sessions[session_id].state
```

Oracle 16: query_key(·)

```
1. define query_key(party, session_id) as:
2.     if session_id ∉ party_states[party].sessions.keys then:
3.         return ⊥
4.     global_sessions[session_id].exposed ← True
5.     return party_states[party].sessions[session_id].S
```

Oracle 17: corrupt(·)

```
1. define corrupt(party, code) as:
2.     for all session ∈ party_states[party].sessions do:
3.         if session.state ≠ ∅ or session.S ≠ ∅ then:
4.             global_sessions[session.session_id].exposed ← True
5.     execute(code) with access to:
6.         • authentic_messages.append(sender = party, ·, ·, ·)
7.         • party_states[party]
```

Some explanation about the variables' purpose and usage is in order. In the following enumeration we mix descriptions of variables and their types.

- session_id, party_id : integer. These identifiers are just integers.
- *party_state* : list of dict mapping session_id to session. This variable is a list containing for each party i a dict called *sessions*, which is a dictionary mapping session_ids to session objects.
- session. This type is a tuple containing the following objects:
  - *key* : session_id. Integer uniquely identifying the session and counterpart-session pair. (In other words, the other party involved in this session has a matching session and it has the same *key*.)
  - A : party_id. This party id indicates the party who is taking on the role of A in the NKA session.
  - B : party_id. This party id indicates the party who is taking on the role of B in the NKA session.
  - *params* : ParSp. This object takes on the value *iparams* as generated by the Init function of the NKA protocol.
  - *state* : StateSp. This variable takes on the value of A_state or B_state in the NKA protocol.
  - *contribution* : ContrSp. This variable takes on the value of this party's contribution in the NKA protocol.

  - $S : \{0,1\}^\ell$. This is the view of the shared noisy key as held by the party in question.
- *global_sessions* : list of global_session. This list of global_session objects contains big picture information on superficial session attributes like the parties involved and whether or not the session has been exposed.
- global_session. This object consists of the following variables:
  - A : party_id. This variable is the party id of the party who assumes the role of A in the NKA session pair.
  - B : party_id. This variable is the party id of the party who assumes the role of B in the NKA session pair.
  - *exposed* : {True, False}. Boolean variable indicating whether the session has been exposed or not.
- *authentic_messages* : list of message objects, representing all information transmitted between parties.
- message. This object consists of the following variables:
  - *sender* : party_id. This variable identifies the originator of the message.
  - *receiver* : party_id. This variable identifies the intended receiver of the message.
  - *session_id* : session_id. This variable identifies the session pair to which this protocol contribution pertains.
  - *contribution* : ContrSp. The actual content of the message: an NKA protocol contribution.
- *test_session_id* : session_id. The identifier of the test session as output by the adversary at the end of the first phase.
- *state* : $\{0,1\}^*$. The adversary's state at the end of the first phase; recording this state allows the adversary to pick up where it left off.
- P_A, P_B : party_id. These identifiers determine the parties involved in the session pair that was chosen as test session by the adversary.
- $S_A, S_B : \{0,1\}^\ell$. These are the views of the noisy session key associated with the two parties in the session pair that was chosen as test session by the adversary.
- $S : \{0,1\}^\ell$. Challenge key, to be fed to the adversary in the second phase. The adversary wins if he can tell whether S was drawn from a uniform distribution or from the intersection of two radius-t spheres centered at S_A and S_B.
- $b, \hat{b} : \{0,1\}$. Bits, one determining whether to sample S at random or from the intersection of spheres; the other being the adversary's guess.

B.3 NK-security and NKD Assumption

Theorem 3. *The NKD Assumption is necessary and sufficient for NK-security.*

This theorem is an immediate corollary of the following two lemmas, both of which have straightforward proofs.

Lemma 5 (NKD $\Longrightarrow$ NK). *Let* A *be a polynomial time quantum adversary in the* NK *game with respect to an NKA protocol* Π *with failure probability* ϵ*, and let* k *and* r *be the number of sessions started and corrupted, respectively, by* A*, and* $\frac{1+\epsilon\frac{k-r}{k}+\frac{r}{k}}{2}+\zeta$ *its winning probability. Then there is a polynomial time quantum algorithm* B *that wins the* NKD *game in polynomial time with probability* $\frac{1+\epsilon}{2}+\frac{\zeta}{k-r}$*.*

Proof. The arguments of B are $(iparams, A\_contr, B\_contr, S)$. B chooses a random session identifier $id \xleftarrow{\$} \{0,\ldots,k-1\}$ and simulates the NK game. The oracles are defined in accordance with Definition 4 except where the session with $session\_id = id$ is concerned. For this session, the instance parameters and both parties' contributions are set to *iparams*, *A_contr*, and *B_contr*. The views of the session keys are set to the same random bitstring of length ℓ.

The adversary $\mathsf{A}_1^{\cdots}(1^\lambda)$ is run and if its output $test\_session\_id \neq k$ then B flips a coin $\hat{b} \xleftarrow{\$} \{0,1\}$ and returns that. Otherwise $\mathsf{A}_2^{\cdots}(state, S)$ is run, where S is B's fourth argument. If session id is exposed, B returns a random coin flip $\hat{b} \xleftarrow{\$} \{0,1\}$ and otherwise B returns the output of $\hat{b} \leftarrow \mathsf{A}_2^{\cdots}$. The exact behavior of B and the modified oracle interface it provides the simulated adversary A with, are presented in Algorithm 18 and oracle $\mathsf{contribute}'$, with the other oracles being identically defined to those in Definition 4.

The tuple (*iparams*, *A_contr*, *B_contr*) associated with each session is identically distributed, including session id. Therefore the probability that $\mathsf{A}_1^{\cdots}$'s output $test\_session\_id = id$ is exactly $1/k$. Let z be shorthand for the output of the NKD game, *i.e.*, $z \leftarrow \mathsf{NKD}^{\mathsf{B}^{\mathsf{A}^{\cdots}}}(1^\lambda)$.

Algorithm 18: $\mathsf{B}^{\mathsf{A}^{\cdots}}(\mathit{iparams}, \mathit{A\_contr}, \mathit{B\_contr}, S)$

```
party_states ← [∅ for all parties]
authentic_messages, global_sessions ← empty_lists
session_counter ← 0
id ←$ {0, ..., k − 1}
test_session_id, state ← A_1^...(1^κ)
if test_session_id ≠ id then:
    return ⊥
b̂ ← A_2^...(state, S)
if global_sessions[id].exposed = True then:
    b̂ ←$ {0, 1}
return b̂
```

Oracle 19: $\mathsf{contribute}'$

```
define contribute(party, session_id) as:
    if session_id ∉ party_states[party].sessions.keys() then:
        return ⊥
    session ← party_states[party].sessions[session_id]
    if party = session.A then:
        session.state, session.contribution ← Π.AContr(session.params)
        if session_id = id then:
            session.contribution ← A_contr
    else:
        session.state, session.contribution ← Π.BContr(session.params)
        it session_id = id then:
            session.contribution ← B_contr
    party_states[party].sessions[session_id] ← session
    msg ← (
        sender = party,
        receiver = {session.A, session.B}\party,
        session_id = session_id,
        contribution = session.contribution)
    authentic_messages.append(msg)
    return msg    ▷ allow adversary to block
```

Then we have:

$$\Pr[\mathsf{NKD}^{\mathsf{B}^{\mathsf{A}^{\cdots}}}(1^\lambda) \not\Rightarrow 0] \stackrel{\triangle}{=} \Pr[z \neq 0] \tag{65}$$

$$= \Pr[z \neq 0 \,|\, z \neq \perp] \cdot \Pr[z \neq \perp] + \Pr[z \neq 0 \,|\, z = \perp] \cdot \Pr[z = \perp] \tag{66}$$

$$\begin{aligned} &= \epsilon + \Pr[z \neq 0 \,|\, z \neq \perp \wedge \mathit{test\_session\_id} = \mathit{id}] \cdot \Pr[\mathit{test\_session\_id} = \mathit{id}] \cdot (1-\epsilon) \\ &\quad + \Pr[z \neq 0 \,|\, z \neq \perp \wedge \mathit{test\_session\_id} \neq \mathit{id}] \cdot \Pr[\mathit{test\_session\_id} \neq \mathit{id}] \cdot (1-\epsilon) \end{aligned} \tag{67}$$

$$= \Pr[\mathsf{NK}^{\mathsf{A}^{\cdots}}(1^\lambda) \not\Rightarrow 0 \,|\, \not\perp] \cdot \frac{1}{k} \cdot (1-\epsilon) + \epsilon + \frac{1}{2} \cdot \frac{k-1}{k} \cdot (1-\epsilon) \tag{68}$$

$$\begin{aligned} &= \Big(\Pr[\mathsf{NK}^{\mathsf{A}^{\cdots}}(1^\lambda) \not\Rightarrow 0 \,|\, \not\perp] \cdot \Pr[\not\perp] \\ &\quad + \Pr[\mathsf{NK}^{\mathsf{A}^{\cdots}}(1^\lambda) \not\Rightarrow 0 \,|\, \perp] \cdot \Pr[\perp] \\ &\quad - \Pr[\mathsf{NK}^{\mathsf{A}^{\cdots}}(1^\lambda) \not\Rightarrow 0 \,|\, \perp] \cdot \Pr[\perp] \Big) \cdot (\Pr[\not\perp])^{-1} \cdot \frac{1}{k} \cdot (1-\epsilon) \\ &\quad + \epsilon + \frac{1}{2} \cdot \frac{k-1}{k} \cdot (1-\epsilon) \end{aligned} \tag{69}$$

$$\begin{aligned} &= \Pr[\mathsf{NK}^{\mathsf{A}^{\cdots}}(1^\lambda) \not\Rightarrow 0] \cdot (\Pr[\not\perp])^{-1} \cdot \frac{1}{k} \cdot (1-\epsilon) \\ &\quad - \left(\frac{r}{k} + \frac{k-r}{k} \epsilon \right) \cdot (\Pr[\not\perp])^{-1} \cdot \frac{1}{k} \cdot (1-\epsilon) \\ &\quad + \epsilon + \frac{1}{2} \cdot \frac{k-1}{k} \cdot (1-\epsilon) \end{aligned} \tag{70}$$

$$\begin{aligned} &= \Pr[\mathsf{NK}^{\mathsf{A}^{\cdots}}(1^\lambda) \not\Rightarrow 0] \cdot \left(\frac{k}{k-r-\epsilon k+\epsilon r} \right) \cdot \frac{1}{k} \cdot (1-\epsilon) \\ &\quad - \left(\frac{r}{k} + \frac{k-r}{k} \epsilon \right) \cdot \left(\frac{k}{k-r-\epsilon k+\epsilon r} \right) \cdot \frac{1}{k} \cdot (1-\epsilon) \\ &\quad + \epsilon + \frac{1}{2} \cdot \frac{k-1}{k} \cdot (1-\epsilon) \end{aligned} \tag{71}$$

$$= \frac{1}{k-r} \Pr[\mathsf{NK}^{\mathsf{A}^{\cdots}}(1^\lambda) \not\Rightarrow 0] - \frac{\epsilon}{k} - \frac{r}{k(k-r)} + \epsilon + \frac{1-\epsilon}{2} \cdot \frac{k-1}{k} \tag{72}$$

$$= \frac{1}{k-r} \left(\frac{1 + \frac{k-r}{k}\epsilon + \frac{r}{k}}{2} + \zeta \right) - \frac{\epsilon}{k} - \frac{r}{k(k-r)} + \epsilon + \frac{1-\epsilon}{2} \cdot \frac{k-1}{k} \tag{73}$$

$$= \frac{\zeta}{k-r} + \frac{\frac{1}{2}}{k-r} + \frac{\epsilon}{2k} + \frac{\frac{r}{2}}{k(k-r)} - \frac{\epsilon}{k} - \frac{r}{k(k-r)} + \epsilon + \frac{1-\epsilon}{2} \cdot \frac{k-1}{k} \tag{74}$$

$$= \frac{1+\epsilon}{2} + \frac{\zeta}{k-r} \tag{75}$$

□

Lemma 6 (NK $\Longrightarrow$ NKD). *Let* A *be a polynomial time quantum adversary in the* NKD *game with respect to an NKA protocol* Π *with failure probability* ϵ*, whose winning probability is* $\frac{1+\epsilon}{2} + \zeta$*. Then there is a polynomial time quantum*

algorithm $\mathsf{B}^{\cdots}$ *that wins the* NK *game with respect to* Π *in polynomial time with probability* $\frac{1+\epsilon}{2} + \zeta$.

Proof. The adversary $\mathsf{B}^{\cdots} = (\mathsf{B}_1^{\cdot\cdot}, \mathsf{B}_2^{\cdot\cdot})$ behaves as follows. In phase 1, $\mathsf{B}_1^{\cdot\cdot}$ starts a session between two random parties and instructs both of them to contribute and converge; he thus obtains $session\_id, iparams, A\_contr, B\_contr$. His output is then $(test\_session\_id = session\_id, state = (iparams, A\_contr, B\_contr))$.

In phase 2, $\mathsf{B}_2^{\cdot\cdot}$ runs on input $(state = (iparams, A\_contr, B\_contr), S)$. He invokes A as an NKD-oracle, namely by passing it the arguments ($iparams$, A_contr, B_contr, S) and obtaining A's guess $\hat{b}$, which is also $\mathsf{B}_2^{\cdot\cdot}$'s output. Whenever A wins, so does $\mathsf{B}^{\cdots}$, so the theorem follows. □

The reduction NKD $\Longrightarrow$ NK loses a security factor $1/(k-r)$, where k is the number of sessions started by the NK adversary and r is the number of sessions corrupted. However, this security loss is a necessary consequence of restricting the number of available sessions to one, as in the NKD game. NK-security and the NKD Assumption remain asymptotically equivalent.