ORAMs in a Quantum World

February 27, 2017

Tommaso Gagliardoni[1], Nikolaos P. Karvelas[2], and Stefan Katzenbeisser[3]

[1] Cryptoplexity, Technische Universität Darmstadt, Germany
`tommaso@gagliardoni.net`

[2] Security Engineering, Technische Universität Darmstadt, Germany
`karvelas@seceng.informatik.tu-darmstadt.de`

[3] Security Engineering, Technische Universität Darmstadt, Germany
`katzenbeisser@seceng.informatik.tu-darmstadt.de`

Abstract. We study the security of *Oblivious Random Access Machines (ORAM)* in the quantum world. First we introduce a new formal treatment of ORAMs, which is at the same time elegant and simpler than the known formalization by Goldreich and Ostrovsky. Then we define and analyze the notion of post-quantum security for ORAMs, i.e., classical ORAMs resistant against quantum adversaries. We show that merely switching to post-quantum secure encryption in a classically secure ORAM construction does not generally yield a post-quantum secure ORAM construction. On the other hand, we provide a post-quantum secure construction based on a modification of Path-ORAM, the most efficient general ORAM construction, introduced by Stefanov et al.
Furthermore, we initiate the study of *Quantum ORAMs (QORAMs)*, that is, ORAM constructions meant to be executed between quantum parties acting on arbitrary quantum data. We address many problems arising when formalizing Quantum ORAMs, and we provide a secure construction (based on Path-ORAM and a quantum encryption scheme introduced by Alagic et al.) which has the interesting property of making read and write operations *inherently equivalent*. In so doing, we develop a novel technique of quantum extractability which is of independent interest. We believe that QORAMs represent a natural and interesting step in the direction of achieving privacy in future scenarios where quantum computing is ubiquitous.

Keywords: Quantum security, Privacy Enhancing Technologies, Oblivious RAM, Path-ORAM

1 Introduction

Since the introduction of Shor's quantum algorithm [40] for solving the discrete logarithm and factoring problems, it has become clear that once scalable quantum computers become available, many of today's most widely used cryptographic tools will become obsolete. In response to the threat of potential future quantum adversaries, new cryptographical constructions [25, 7, 4] have been proposed, that are based on mathematical problems which are believed to be quantum-hard [32, 35, 13].

Although encryption schemes are an important building block for many of the existing cryptographic tools, more complex architectures are built by combining various cryptographic primitives. In such cases, it is natural to ask if merely replacing the underlying primitives with their quantum-secure counterparts would yield quantum-secure architectures. Various recent works [2, 12, 48] have found a negative answer to such question: not only this is not the case for many known constructions, but also whole families of security proofs can be identified, which do not hold in a quantum setting.

Under this light, an interesting direction is the problem of building a quantum-secure *Oblivious Random Access Machine (ORAM)*. Oblivious RAM was introduced in the early '90s by Goldreich and Ostrovsky in [20] as a mean of software protection. It has since received an increasing deal of attention from the cryptographic community and has been used as building block for many applications that include secure processors [33], storage volumes hiding [9], oblivious storage [21] and privacy preserving health records [34]. With ongoing constant improvements in the research towards a functional quantum computer, it is therefore reasonable to wonder if and under which conditions such privacy enhancing technologies and architectures would still remain secure in a quantum world.

A major obstacle to tackle this problem is given by the fact that the ORAM definition given by Goldreich and Ostrovsky is rather involved. Although this is usually a good thing, given the level of formalism required in modern cryptography, and especially for such high-level functionalities such as ORAM, the drawback is an increased difficulty met when analyzing the security of a given construction. In the field of provable security in particular, this has shown to be detrimental for the efforts of the community to analyze even the most basic ORAM constructions. In other words, the complex formalism given in the original formulation makes very hard to prove a certain ORAM secure; in fact, all the existing proofs we are aware of [46, 38, 47, 43, 39, 42] are rather 'sketchy', and do not fully respect that formalism anyway. We believe this to be a substantial problem: When excessive formalism stands on the way that leads to a useful treatment of the theory for real-world applications, it is probably necessary to find a compromise between excessive and insufficient formalism. Especially in the field of security, this approach is often the best solution.

1.1 Our Results

In this paper, we provide many contributions to the study of ORAM security, both in the classical and in the quantum world.

First of all, we develop and present a new formal model for describing ORAMs, simpler than the one originally proposed by Goldreich and Ostrovsky (their definition spans through a multitude of pages), yet elegant and elastic enough to cover most of the existing interesting constructions in a rigorous way. In particular, we model the parties involved in an ORAM protocol as BPP circuits, defining carefully the meaning of *database*, *data request*, and *communication transcript* in terms of circuit registers.

Then we define a novel, game-based security model for ORAMs, which is very flexible and at least as strong as all the other models proposed to date. We start by defining *access patterns*, which are the adversarial views produced during an execution of data requests. Our adaptive indistinguishability notion states that, for any computationally bounded adversary, it should be difficult to distinguish between the execution of different data requests by looking at the resulting access patterns, even by allowing the adaptive execution of polynomially many learning queries. To our knowledge, this is the first time that these notions are defined in a homogeneous framework that allows for a formal treatment of any ORAM construction, without going to formal extremes that make it useless in practice.

Our new model allows us to argue about the extension of ORAM security to the quantum setting in a straightforward manner. To our knowledge, this is the first work where the quantum security of ORAMs is analyzed. In line with most of the existing ORAM security scenarios, we consider honest-but-curious adversaries, which are able to interact classically with the scheme, but can perform local quantum computation. Our model therefore falls in the category of so-called *'post-quantum' security* [8].

In particular, as every ORAM scheme uses some sort of encryption in order to protect the privacy of the storage, it is interesting to relate the security of an ORAM construction to the security of the encryption scheme it uses. Because ORAM constructions can rely on many other different primitives in addition to encryption schemes, it is not reasonable to expect that post-quantum security can be achieved by merely switching to underlying post-quantum encryption. As a counterexample, we show that Path-ORAM [43], the most efficient general ORAM construction, can be attacked by a polynomially bounded quantum adversary by exploiting the weakness of a not-carefully-chosen PRNG, even if the underlying encryption primitive is post-quantum secure.

On the other hand, we show that Path-ORAM, if instantiated using *both* a post-quantum secure encryption scheme *and* a post-quantum secure PRNG, indeed achieves security against quantum adversaries. This result is important from a practical perspective, as it shows an easy way to build efficient post-quantum ORAMs in a completely black-box way.

Furthermore, we go beyond the post-quantum model of security by initiating the study of *Quantum ORAMs (QORAMs)*, that is, ORAM constructions meant to be executed between quantum parties acting on arbitrary quantum data. We motivate this study by noting that, once quantum computers start being available, it would be natural to expect that many parties involved in an ORAM protocol would need to act on quantum data, i.e., *qubits*. This motivation is further strengthened by the practical consideration that the first commercially available quantum computers would be very expensive, and server-client models are likely to appear, where computationally limited quantum clients will interact with more powerful quantum servers.

This scenario presents many unique challenges that have no counterpart in the classical setting. One outstanding problem in quantum reductions is to define a notion of 'honest-but-curious' adversary, or in general 'soundness-preserving' adversary. As measuring an unknown quantum state disturbs it with high probability, and because of the No-Cloning Theorem, it is usually impossible to give a notion of 'read-only' for quantum channels. Therefore, in all those applications where the quantum adversary is allowed to extract information from a physical system which is part of the protocol in exam, one has often to go to great lengths in order to show that the resulting disturbance is tolerable for the protocol itself. This is usually done in many different, ad-hoc, and often complex ways, depending on the scenario [10, 12, 11, 3].

In our work we formalize this problem by introducing the notion of a *safe extractor*. This can be seen as a 'subroutine' of the adversary, which extracts some amount of quantum information from a certain quantum register, but such that its action on the register is 'computationally undetectable'. We believe this very general technique to be of independent interest, and applicable to many other scenarios.

Thanks to the flexibility and rigor of our new formalism developed for classical ORAMs, we can then define QORAMs in a precise way, and we give a sound treatment of their security. Finally, we provide a QORAM construction, based on Path-ORAM and a quantum encryption scheme introduced in [1], which has the interesting property of making read and write operations *inherently equivalent*.

1.2 Related Work

Oblivious RAM was introduced in the '90s by Goldreich and Ostrovsky in their seminal work [20] as a means to protect software from piracy. Since then, ORAM has received a lot of attention from the cryptographic community [22, 38, 46, 42, 45, 39, 43, 18, 17], and the constructions proposed are becoming more and more efficient.

An ORAM can be thought of in a client-server model, where the client outsources to a server data blocks of fixed size, and wants to access them afterwards in a way that does not reveal which blocks have been accessed. In [20] Goldreich and Ostrovsky propose a construction, where the client's data is stored in a pyramid-like structure. The heavy communication cost of such a construction left much space for further improvements in the original scheme, which were made by subsequent works like [38, 46]. In [29], privacy leakage were identified in most of the solutions proposed so far, and countermeasures were proposed.

The ORAM landscape changed radically after the introduction of tree-based ORAMs, by Shi et al. [42]. In these solutions the client's data blocks are stored in a binary tree and every block is randomly associated with a leaf of the data structure. Perhaps the most intuitive approach in tree-based ORAM contructions was proposed by Stefanov et. al. in their Path-ORAM solution [43]. Recently Garg et. al. [17], used Path-ORAM to further improve on the communicational complexity of ORAM protocols.

In a concurrent and independent work, Garg et al. [17] proposed a new model for describing ORAMs in a formal way. Although their model is a definitive improvement to the current ORAM landscape, still it omits some details which are important when analyzing the security of ORAMs in a quantum world. In particular, we show their security model to be equivalent to ours. However, their model is simulation-based (ours is game-based), and furthermore it does not allow to argue about the relation between the post-quantum security of an ORAM construction and that of its underlying building blocks. Moreover, their definition of ORAM could not be easily adapted to define Quantum ORAMs.

2 Preliminaries

In the rest of this paper we will use the term 'classical' for 'non-quantum', $n \in \mathbb{N}$ as the security parameter, and $\perp$ as a special symbol denoting an error or lack of meaningful information. BPP stands for 'bounded probabilistic polynomial time', and by 'algorithm' we mean a uniform family of circuits; therefore 'BPP algorithm' stands for 'uniform family of poly-sized Boolean circuits with randomness'. Given a security *game*, or *experiment*, the output 1 denotes success (winning condition), while 0 denotes failure (loss condition). For the definitions of *secret-key encryption scheme* (SKE) $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, *classical indistinguishability of ciphertexts game* $\mathsf{Game}^{\mathtt{IND}}_{\mathcal{A},\mathcal{E}}(n)$, and *Discrete Logarithm Problem (DLP)*, we point the reader to [19]. We only recall here the standard definition of IND-CPA security for symmetric-key encryption schemes.

Definition 1 (IND-CPA)**.** *A SKE* $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *has* indistinguishable ciphertexts under chosen plaintext attack *(or, it is IND-CPA-secure) iff for any BPP algorithm* $\mathcal{A}$ *with oracle access to* Enc*:*

$$\left| \Pr\left[\mathsf{Game}^{\mathit{IND}}_{\mathcal{A}^{\mathsf{Enc}},\mathcal{E}}(n) = 1\right] - \frac{1}{2} \right| \leq \mathit{negl}(n).$$

Intuitively, a SKE is IND-CPA-secure if no computationally bounded adversary can reliably distinguish between the encryption of two plaintexts of his choice even if allowed to adaptively learn polynomially many other encryptions.

2.1 Quantum Security Models

We refer to [37] for commonly used notation and quantum information-theoretic concepts. BQP stands for 'bounded probabilistic quantum polynomial time'; therefore 'BQP algorithm' stands for 'uniform family of poly-sized quantum circuits'. We will denote by $\mathcal{H}_d$ a Hilbert space of dimension 2^d. We will denote pure states with ket notation, e.g., $|\varphi\rangle$, while mixed states will be denoted by lowercase Greek letters, e.g. ρ. The set of positive, trace-preserving bounded operators on $\mathcal{H}$ (that is, the set of all possible mixed states on $\mathcal{H}$) will be denoted by $\mathfrak{D}(\mathcal{H})$, while $\|.\|_{Tr}$ is the *trace norm*, and $\|.\|_{\diamond}$ is the *diamond norm*. The *computational base* for a Hilbert space $\mathcal{H}_d$ is the set of orthonormal vectors denoted by $\{\ |0\dots00\rangle,\ |0\dots01\rangle, \dots,\ |1\dots11\rangle\ \}$.

Post-Quantum Security for Encryption

We first recall the notion of *post-quantum security* [8]. The idea is to define, for a given cryptographic primitive, notions of security which are supposed to remain meaningful even in a world where quantum computers have been introduced and are available to the adversary. That is, post-quantum security is about the security of *classical* primitives *after* (hence 'post-') quantum computing becomes available. The model considered is where an adversary can interact classically (i.e., 'in a normal way') with the given primitive, or oracle, but additionally has access to a quantum computing device for performing extra computation not normally available to classical adversaries. It is important to notice that in the last few years many new scenarios have been proposed, where this 'canonical' way of interpreting post-quantum security has been challenged, by giving additional power and resources to the quantum adversary, see for example [10, 27, 36, 12, 11, 16, 26, 3, 48, 30, 31, 15]. In this work however, given the security model traditionally used in the study of ORAMs, we will only consider post-quantum security as in the canonical meaning. We recall here what it means for a classic SKE to be post-quantum secure in this sense.

Definition 2 (pq-IND-CPA)**.** *A SKE* $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *has* post-quantum indistinguishable ciphertexts under chosen plaintext attack *(or, it is pq-IND-CPA-secure) iff for any BQP algorithm* $\mathcal{A}$ *with oracle access to* Enc*:*

$$\left| \Pr\left[\mathsf{Game}^{\mathit{IND}}_{\mathcal{A}^{\mathsf{Enc}},\mathcal{E}}(n) = 1 \right] - \frac{1}{2} \right| \leq \mathit{negl}(n).$$

Quantum Encryption

In this work we also consider *quantum primitives* [14], that is, cryptographic functionalities meant to be natively realized through a quantum computer, by acting on quantum information (qubits). Here the oracles themselves are usually not classical, but defined as operations $\mathcal{O} : \mathfrak{D}(\mathcal{H}) \to \mathfrak{D}(\mathcal{H})$ acting directly on qubits.

We recall here the definition of *symmetric-key quantum encryption scheme* (QSKE) from [1].

Definition 3 (QSKE)**.** *A* quantum symmetric-key encryption scheme (QSKE) $\mathcal{E}_Q$ *with m-qubit plaintexts and c-qubits ciphertexts is a tuple* $(Q.\mathsf{KeyGen}, Q.\mathsf{Enc}, Q.\mathsf{Dec})$ *of BQP algorithms:*

1. *(key generation)* $Q.\mathsf{KeyGen} : 1^n \mapsto k \in \mathcal{K}$
2. *(encryption)* $Q.\mathsf{Enc} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$
3. *(decryption)* $Q.\mathsf{Dec} : \mathcal{K} \times \mathcal{Y} \to \mathcal{X}$

such that $\| Q.\mathsf{Dec} \circ Q.\mathsf{Enc} - \mathbb{I}_{\mathcal{X}} \|_{\diamond} \leq \mathit{negl}(n)$ *for all* $k \in \mathit{Supp}\left(Q.\mathsf{Gen}(1^n)\right)$*, where* $\mathcal{K} = \{0,1\}^{\mathit{poly}(n)}$ *is the* key space*,* $\mathcal{X} \subset \mathfrak{D}(\mathcal{H}_m)$ *is the* plainstate space*,* $\mathcal{Y} \subset \mathfrak{D}(\mathcal{H}_c)$ *is the* cipherstate space*,* $\mathbb{I}$ *is the identity operator, and* $Q.\mathsf{Dec}, Q.\mathsf{Enc}$ *must be intended acting with the same (classical) key* k*.*

Also in [1], security definitions for such QSKEs are given. Here we only consider the following.

Definition 4 ($\mathsf{Game}^{\mathtt{Q\text{-}IND}}_{\mathcal{A},\mathcal{E}_Q}(n)$)**.** *Let $\mathcal{E}_Q = (Q.\mathsf{KeyGen}, Q.\mathsf{Enc}, Q.\mathsf{Dec})$ be a QSKE, and $\mathcal{A}$ (the Adversary) a BQP algorithm. The* indistinguishability of quantum ciphertexts game $\mathsf{Game}^{\mathtt{Q\text{-}IND}}_{\mathcal{A},\mathcal{E}_Q}(n)$ *proceeds as follows:*

1. $\rho_0, \rho_1, \sigma \leftarrow \mathcal{A}(1^n)$ *(σ is an internal memory state used by the adversary);*
2. $k \leftarrow Q.\mathsf{Gen}(1^n), b \xleftarrow{\$} \{0,1\}$*;*
3. $\psi \leftarrow Q.\mathsf{Enc}_k(\rho_b)$*, while ρ_{1-b} is traced out;*
4. $b' \leftarrow \mathcal{A}(\psi, \sigma)$.

$\mathcal{A}$ wins the game iff $b = b'$.

Definition 5 (Q-IND-CPA)**.** *A QSKE $\mathcal{E}_Q = (Q.\mathsf{KeyGen}, Q.\mathsf{Enc}, Q.\mathsf{Dec})$ has* indistinguishable quantum ciphertexts under chosen quantum plaintext attack *(or, it is Q-IND-CPA-secure) iff for any BQP algorithm $\mathcal{A}$ with quantum oracle access to $Q.\mathsf{Enc}$:*

$$\left| \Pr\left[\mathsf{Game}^{\mathtt{Q\text{-}IND}}_{\mathcal{A}^{Q.\mathsf{Enc}},\mathcal{E}_Q}(n) = 1 \right] - \frac{1}{2} \right| \leq negl(n).$$

2.2 Pseudorandom Number Generators

In this section we briefly recall security notions for *pseudorandom number generators.*

Definition 6 (PRNG, pq-PRNG)**.** *Let ℓ be a polynomial such that $\ell(n) \geq n + 1\ \forall n \in \mathbb{N}$. A* pseudorandom number generator, or PRNG (resp., post-quantum pseudorandom number generator, or pq-PRNG) *with expansion factor $\ell(.)$ is a deterministic polynomial-time algorithm $\mathcal{G} = \mathcal{G}_\ell$ such that:*

1. *given as input a bitstring s of length n, (the* seed*), outputs a bitstring $\mathcal{G}(s)$ of length $\ell(n)$; and*
2. *for any BPP (resp., BQP) algorithm $\mathcal{D}$:*
$$\left| \Pr\left[\mathcal{D}(r) = 1 \right] - \Pr\left[\mathcal{D}(\mathcal{G}(s)) = 1 \right] \right| \leq negl(n),$$
where $r \xleftarrow{\$} \{0,1\}^{\ell(n)}, s \xleftarrow{\$} \{0,1\}^n$, and the probabilities are over the choice of r and s, and the randomness of $\mathcal{D}$.

Clearly, a pq-PRNG it is also a PRNG. However, the opposite is not believed to hold, as the following example shows.

Lemma 7. *Under the DLP hardness assumption, there exists a PRNG $\mathcal{G}^*$ which is* quantumly predictable*. I.e., there exists a non-negligible function ν and a BQP algorithm $\mathcal{D}$ which, on input n sequential values output by $\mathcal{G}^*$ on any random seed, predicts the $(n+1)$-th value output by $\mathcal{G}^*$ with probability at least $\nu(n)$.*

The proof of the above lemma can be found in Appendix A.1.

3 The New ORAM Model

In this section we recall the concept of classical *Oblivious Random Access Machine (ORAM)*, and we define and analyze security models against classical and quantum adversaries. Defining ORAMs in a fully formal way is a delicate and strenuous task [20]. Therefore, in the following we will introduce a simplified model which covers all existing ORAM constructions without delving too much into the fine print - but still retaining a reasonable level of formalism - and which has the advantage of being much easier to treat. We believe our model will prove to be a valuable tool in the formal analysis of existing ORAM constructions, which is an aspect too often overlooked.

Informally, an ORAM is an interactive protocol between two parties: a *client* $\mathcal{C}$ and a *server* $\mathcal{S}$, which we model as two BPP Turing machines (or, in our case, uniform families of circuits) sharing a communication tape (circuit register) $\varXi$. In this scenario, a computationally limited $\mathcal{C}$ wants to outsource a *database (DB)* to the more powerful $\mathcal{S}$. Moreover, $\mathcal{C}$ wants to perform *operations* on the DB (by interactively communicating with $\mathcal{S}$) in such a way that $\mathcal{S}$, or any other honest-but-curious adversary $\mathcal{A}$ having read-only access to $\varXi$ and $\mathcal{S}$'s internal memory, cannot determine the nature of such operations. The security notion of an ORAM protocol is hence a particular notion of *privacy.*

More formally: we define *blocks*, the basic storage units used in an ORAM construction. A block is an area of memory (circuit register) storing a B-bit value, for a fixed parameter $B \in \mathbb{N}$ which depends on $\mathcal{C}$'s and $\mathcal{S}$'s architectures. A *database* (DB) of size $N \in \mathbb{N}$ is an area of $\mathcal{S}$'s memory which stores an array $(\mathtt{block}_1, \dots, \mathtt{block}_N)$ of such blocks. As we assume this database to reside on the server's side, we will denote it as $\mathcal{S}.\mathtt{DB}$. Notice that the precise way this array of blocks is represented in the database is unspecified, and left to the exact implementation of the ORAM scheme taken into account. For example, in PathORAM, the server's database $\mathcal{S}.\mathtt{DB}$ stores blocks in a binary tree structure. We will abuse notation and write that $\mathcal{S}.\mathtt{DB}(i) = \mathtt{block}$ if $\mathtt{block}$ is the i-th component of $\mathcal{S}.\mathtt{DB}$, and that $\mathtt{block} \in \mathcal{S}.\mathtt{DB}$ if $\mathtt{block}$ is stored at some position in the database $\mathcal{S}.\mathtt{DB}$.

Next we define *data units* as the basic units of data that the client wants to access, read, or write. Formally, a data unit is a D-bit value for a fixed parameter $D \leq B$ which depends on $\mathcal{C}$'s and $\mathcal{S}$'s architectures. Every block encodes (usually in an encrypted form) a data unit, plus possibly additional auxiliary information such as a block identifier, checksum, or hash value. Since every block can encode a single data unit, at any given time t it is defined a function $\mathtt{Data}_t : \mathcal{S}.\mathtt{DB} \to \{0,1\}^D$. With abuse of notation, we will denote by $\mathtt{Data}(\mathtt{block})$ the data unit encoded in the block $\mathtt{block}$ at a certain time. The client $\mathcal{C}$ can operate on the database through *data requests.*

Definition 8 (Data Request)**.** *A* data request *to a database $\mathcal{S}.\mathtt{DB}$ of size N is a tuple $\mathtt{dr} = (\mathtt{op}, i, \mathtt{data})$, where $\mathtt{op} \in \{read, write\}, i \in \{1, \dots, N\}$, and $\mathtt{data} \in \{0,1\}^D$ is a data unit ($\mathtt{data}$ can also be $\perp$ if $\mathtt{op} = read$).*

Finally, we define the meaning of a *communication transcript* during an execution of an ORAM protocol. Since this also depends on the exact implementation of the ORAM scheme, we will use the following definition.

Definition 9 (Communication Transcript). *A communication transcript $\mathtt{com}_t$ at time t is the content of the communication channel Ξ at time t of the protocol's execution.*

Notice that the above defines the communication transcript as a function of time, but since an ORAM is a multi-round interactive protocol we will just consider $\mathtt{com}$ as a discrete function of the round $1, 2, \ldots$ of the protocol.

We are now ready to give a definition of ORAM. We assume that a server's database is always initialized empty (usually with randomized encryptions of 0 elements as blocks), and it is left up to the client the task of 'populating' the database with appropriate *write* operations.

Definition 10 (ORAM). *Let $\widetilde{N} \in \mathbb{N}, M \geq D$ and $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a secret-key encryption scheme mapping M-bit plaintexts to B-bit ciphertexts. An* ORAM *$\mathtt{ORAM}_{\mathcal{E}}$ with parameters $(D, \widetilde{N}, \mathcal{E})$ is a pair of two-party interactive randomized algorithms, $(\mathtt{ORAM.Init}, \mathtt{ORAM.Access})$, such that:*

- *$\mathtt{ORAM.Init}(n, N) \to (\mathcal{C}, \mathcal{S})$ in the following way:*
  1. *n is the security parameter, $N < \widetilde{N}$;*
  2. *$k \leftarrow \mathsf{KeyGen}(1^n)$ is generated by $\mathcal{C}$;*
  3. *$\mathcal{S}$ includes a database $\mathcal{S}.\mathtt{DB} = (\mathtt{block}_1, \ldots, \mathtt{block}_N)$;*
- *$\mathtt{ORAM.Access}(\mathcal{C}, \mathcal{S}, \mathtt{dr}) \to (\mathcal{C}', \mathcal{S}', \mathtt{com})$ in the following way:*
  1. *$\mathcal{C}$ issues a data request $\mathtt{dr}$;*
  2. *$\mathcal{C}$ and $\mathcal{S}$ communicate through Ξ and produce the communication's transcript $\mathtt{com}$;*

One might wonder why it is necessary to explicitly condition the definition of an ORAM in respect to a symmetric-key encryption scheme $\mathcal{E}$. It is actually possible to use different primitives, such as public-key encryption or functional encryption, but most of the known ORAM constructions work well with just a simple primitive such as symmetric-key encryption. One might also wonder why the definition does not depend on other cryptographic primitives, such as PRNGs or hash functions. The reason is that not all ORAM constructions use such primitives, for example the 'trivial' ORAM scheme [20] (which consists in just transferring the whole encrypted database from $\mathcal{S}$ to $\mathcal{C}$ and back at every data request) does not use anything else than a symmetric-key encryption scheme $\mathcal{E}$. We leave the extension of our model to the general case as a possibility for future work. On the other hand, notice that encryption of the database is a minimal requirement for security, as we will see, therefore it makes sense to explicitly specify the scheme $\mathcal{E}$ in the notation.

An ORAM must satisfy *soundness* and *security*. We are going to define security in Section 3.1. Regarding soundness, the exact specification depends on the particular ORAM construction considered. A simplified, game-based definition of soundness (*'correctness'*) can be found in [17], but it is difficult to adapt to our model (which is more aimed at studying ORAM security), while a general definition (that can be found in [20]) is rather involved, and goes outside the scope of this work. The meaning of the soundness property is that the ORAM protocol 'should work', i.e., after any execution of `ORAM.Init` or `ORAM.Access` the two parties $\mathcal{C}$ and $\mathcal{S}$ must be left in such a state that allows them to continue the protocol in the next round. Despite the generality of this statement, in our model minimal soundness conditions can be identified, which must hold for *any* ORAM construction.

Definition 11 (Minimal ORAM Soundness Conditions)**.** *An ORAM construction* $\mathtt{ORAM}_{\mathcal{E}}$ *has* minimal soundness *if the following hold:*

1. *for any* (n, N)*, if* $(\mathcal{C}, \mathcal{S}) \leftarrow \mathtt{ORAM.Init}(n, N)$*, then the secret key* k *from Def. 10 must be* accessible *by* $\mathcal{C}$*;*
2. *for any* $\mathtt{dr} = (\mathtt{op}, i, \mathtt{data})$*, if* $(\mathcal{C}', \mathcal{S}', \mathtt{com}) \leftarrow \mathtt{ORAM.Access}(\mathcal{C}, \mathcal{S}, \mathtt{dr})$*, then:*
   (a) *if the secret key* k *is accessible by* $\mathcal{C}$*, then* k *is also* accessible *by* $\mathcal{C}$*';*
   (b) *if* $\mathtt{op} = read$ *and* $\mathcal{S}.DB(i) = \mathtt{block}$*, then* $\mathtt{Data}(\mathtt{block})$ *must be* accessible *by* $\mathcal{C}$*';*
   (c) *if* $\mathtt{op} = write$ *and* $\mathcal{S}'.DB(i) = \mathtt{block}$*, then* $\mathtt{Data}(\mathtt{block}) = \mathtt{data}$*.*

By 'accessible' we mean the following: an element x is *accessible* by a circuit M, if M can simulate the constant oracle $\mathcal{O}_x(.) \mapsto x$. In particular, since we only deal with poly-time algorithms, this simulation must be poly-time. The reason of this definition is that we want to express the fact that a certain data is available to a certain algorithm, but the way this is implemented can depend on the particular ORAM construction. For example, we could require that data is stored somewhere in the circuit's registers, but it could also be the case that the data is stored on some other auxiliary register, or that it is stored in an encrypted form, and the decryption key itself is stored somewhere else. We do not care about the specific case, as long as the circuit has an efficient way to retrieve and operate on that data when needed. This is the intuition behind the use of an oracle. Moreover, in Definition 11, notice that conditions 1 and 2a do not say anything about $\mathcal{S}$ having access to the key k or not: This is a property of *security*, not soundness, as we will see in Section 3.1.

An ORAM scheme `ORAM` can have additional soundness conditions, depending on the particular construction. Moreover, $\mathcal{C}$ (resp., $\mathcal{S}$) can of course perform additional computations, and be modified during the execution of the protocol to $\mathcal{C}$' (resp., $\mathcal{S}$') not only through `ORAM.Access` calls, but also between consecutive calls of `ORAM.Access`. The only requirement is that all the soundness conditions (the minimal ones above as well as the special ones) must always be satisfied. When this happens, we say that $\mathcal{C}$' is a *sound evolution* of $\mathcal{C}$ and that $\mathcal{S}$' is a *sound evolution* of $\mathcal{S}$.

3.1 Classical and Post-Quantum Security

We now look at the security model for ORAMs against classical and quantum adversaries. Traditionally, the threat model in this case is defined by an *honest-but-curious adversary* $\mathcal{A}$. This means that $\mathcal{A}$ is some entity who wants to compromise $\mathcal{C}$'s privacy by having access to the communication channel Ξ and $\mathcal{S}$'s internal memory, but who is not allowed to modify the content of the channel against the protocol, i.e., soundness must be preserved. In general, one does not lose generality by assuming that $\mathcal{S}$ itself is the adversary: $\mathcal{S}$ must behave 'honestly' (in the sense that he follows the protocol, in particular related to the protocol's soundness), but at the same time he will use all the information he can get through the interaction with $\mathcal{C}$ in order to compromise $\mathcal{C}$'s privacy. In particular, this also implies that $\mathcal{S}$ cannot know the key k generated during `ORAM.Init`, as noted above.

Formally, this model is defined in terms of *access patterns*, which are the adversarial views during an execution of data requests in `ORAM.Access`. Security requires that the adversary's view over a certain run of the protocol does not leak any information about the data requests executed by $\mathcal{C}$. This formulation reminds of the definition of *semantic security* for encryption schemes [19]. As in that case, equivalent but easier-to-deal-with formulations can be given in terms of *computational indistinguishability of access patterns* [4]. In this work, we will consider an adaptive, game-based indistinguishability notion stating that for any two data requests, no computationally bounded adversary with knowledge of the access pattern of the client executing one of the two can distinguish which one was executed. Next, we show that this novel definition is equivalent to the simulation-based notion given in [17], which states that no computationally bounded adversary can distinguish between the interaction with a real client and the interaction with a simulator that produces bogus transcripts.

More formally: when a data request is executed, we assume that the honest-but-curious adversary $\mathcal{A}$ records all the communication between $\mathcal{C}$ and $\mathcal{S}$, plus the changes in $\mathcal{S}$'s internal status. Without loss of generality, as we assume that $\mathcal{A}$ and $\mathcal{S}$ coincide, we assume that the only meaningful changes in respect to $\mathcal{A}$ only happen in the database area, $\mathcal{S}.\mathtt{DB}$, between the beginning and the end of an `ORAM.Access` execution. The communications are polynomially bounded and, for simplicity, we assume that the channel Ξ does not erase symbols, i.e., it is write-once. Hence, the adversarial view is composed of the communication transcript, and the server's database before and after the execution of the data request. We :make call this adversarial view, the *access pattern* of the execution.

Definition 12 (Access Pattern)**.** *Given ORAM client and server $\mathcal{C}$ and $\mathcal{S}$, and a data request $\mathtt{dr}$, the* access pattern $\mathtt{ap}(\mathtt{dr})$ *is the tuple* $(\mathcal{S}.\mathtt{DB}, \mathtt{com}, \mathcal{S}'.\mathtt{DB})$*, where* $(\mathcal{C}', \mathcal{S}', \mathtt{com}) \leftarrow \mathtt{ORAM.Access}(\mathcal{C}, \mathcal{S}, \mathtt{dr})$*.*

Next, we define formally a *classical* (resp. *quantum*) *ORAM adversary.*

Definition 13 (Classical and Quantum ORAM Adversary)**.** *A* classical *(resp.* quantum*)* ORAM adversary $\mathcal{A}$ *is a BPP (resp. BQP) algorithm which has complete control of* $\mathcal{S}$*, as long as the ORAM's soundness is preserved.*

[4] The stronger notion of *statistical indistinguishability of access patterns* can also be given, but we will not address it in this work.

Notice that we only take into account computationally bounded adversaries. We define the security of an ORAM through the following indistinguishability game.

Definition 14 ($\mathsf{Game}^{\texttt{AP-IND-CQA}}_{\mathcal{A},\texttt{ORAM}}(n)$)**.** *Let $\texttt{ORAM} = (\texttt{ORAM.Init}, \texttt{ORAM.Access})$ be an ORAM construction, n a security parameter and $\mathcal{A}$ an ORAM adverary. The* computational indistinguishability of access patterns game under adaptive chosen query attack $\mathsf{Game}^{\texttt{AP-IND-CQA}}_{\mathcal{A},\texttt{ORAM}}(n)$ *proceeds as follows:*

1. *$\mathcal{A}$ chooses $N \leq \widetilde{N}$;*
2. *$(\mathcal{C}, \mathcal{S}) \leftarrow \texttt{ORAM.Init}(n, N)$;*
3. *first CQA learning phase: for $i = 1, \ldots, q_1 \in \mathbb{N}$, $\mathcal{A}$ repeats (adaptively) the following:*
   (a) *$\mathcal{A}$ chooses a data request $\texttt{dr}_i$;*
   (b) *$\mathcal{C}$ executes $\texttt{ORAM.Access}$ on $\texttt{dr}_i$;*
   (c) *$\mathcal{A}$ receives $\texttt{ap}(\texttt{dr}_i)$;*
4. *challenge phase: $\mathcal{A}$ chooses two data request $\texttt{dr}^0$ and $\texttt{dr}^1$;*
5. *$\mathcal{C}$ flips a random secret bit $b \xleftarrow{\$} \{0,1\}$ and executes $\texttt{ORAM.Access}$ on $\texttt{dr}^b$;*
6. *$\mathcal{A}$ receives $\texttt{ap}(\texttt{dr}^b)$;*
7. *second CQA learning phase: for $j = 1, \ldots, q_2 \in \mathbb{N}$, $\mathcal{A}$ repeats (adaptively) the following:*
   (a) *$\mathcal{A}$ chooses a data request $\texttt{dr}_j$;*
   (b) *$\mathcal{C}$ executes $\texttt{ORAM.Access}$ on $\texttt{dr}_j$;*
   (c) *$\mathcal{A}$ receives $\texttt{ap}(\texttt{dr}_j)$;*
8. *$\mathcal{A}$ outputs a bit b'.*

$\mathcal{A}$ wins the game iff $b = b'$.

Notice that, since $\mathcal{A}$ is polynomially bounded, q_1 and q_2 are at most polynomials in n. We are now ready to define the classical and post-quantum security notions that we will use in the rest of the paper.

Definition 15 ((Post-Quantum) Access Pattern Indistinguishability Under Adaptive Chosen Query Attack)**.** *An ORAM construction $\texttt{ORAM}$ has* computationally indistinguishable access patterns under adaptive chosen query attack *(or, it is AP-IND-CQA-secure) iff for any classical ORAM adversary $\mathcal{A}$:*

$$\left| \Pr\left[\mathsf{Game}^{\texttt{AP-IND-CQA}}_{\mathcal{A},\texttt{ORAM}}(n) = 1\right] - \frac{1}{2} \right| \leq negl(n).$$

Furthermore, $\texttt{ORAM}$ has post-quantum computationally indistinguishable access patterns under adaptive chosen query attack *(or, it is pq-AP-IND-CQA-secure) iff the above also holds for any quantum ORAM adversary.*

Clearly, if an ORAM is pq-AP-IND-CQA-secure, then it is also AP-IND-CQA-secure. The converse does not hold (under standard hardness assumptions) as we will see.

3.2 Advantages of our Security Model

Traditionally, in the ORAM literature, security has been considered in terms of indistinguishability of access patterns, as we do in this work, but in a non-adaptive fashion. That is, the adversary chooses two different data requests tuples of the same length, and has to distinguish between the access patterns produced by the client executing one of the two, chosen at random. Our model has the advantage that it also considers adversaries who play this game changing adaptively their sequences of data requests.

In a concurrent and independent work [17], Garg et al. introduced another, simulation-based security definition. Their definition states that for any ORAM adversary, it must be computationally hard to distinguish between the access pattern distributions produced by a real client and by a simulator producing bogus transcripts, even if the adversary is allowed to choose adaptively the data requests to be executed by the real client. The original definition in [17] does not take into account quantum adversaries, but can be easily extended in that sense. Readapting this definition to our detailed formalism, we obtain the following.

Definition 16 (Access Pattern Simulability Under Adaptive Chosen Query Attack)**.** *An ORAM construction* $\mathtt{ORAM}$ *has* simulable access patterns under adaptive chosen query attack *(or, it is AP-SIM-CQA-secure) iff for any classical ORAM adversary $\mathcal{A}$ the following two distributions are computationally indistinguishable:*

1. $\mathtt{ap}(\mathtt{dr} \leftarrow \mathcal{A})$;
2. $\mathtt{ap}(\mathtt{dr} \xleftarrow{\$} \{\text{'read'},\text{'write'}\} \times \{1,\ldots,N\} \times \{0,1\}^D)$.

We show in Appendix A.2 that the two models are actually equivalent.

Theorem 17. *An ORAM construction* $\mathtt{ORAM}$ *is AP-SIM-CQA secure iff it is AP-IND-CQA secure.*

The idea of the proof is to go through a hybrid argument in the same way that shows IND-CPA security for encryption schemes to be equivalent to Real-or-Random security (see for example [5]).

Our security notion AP-IND-CQA is therefore at least as strong as all other security notions for ORAM introduced to date, but it has the advantage of being game-based (and hence easier to deal with in security reductions). Furthermore, as we will see in Section 5.1, it translates more smoothly to a 'fully quantum' scenario, where the concepts of access patterns and oracle access have to be phrased carefully.

3.3 PathORAM

As a first application of our new formalism (but also because we will make heavy use of it through the rest of the paper), we recall here PathORAM, one of the most efficient ORAM constructions proposed to date, introduced by Stefanov at al. in [43]. We only give a high-level explanation of PathORAM, and for a thorough description of the construction, as well as a detailed proof of its functionality, we refer to [41].

In PathORAM a client stores N blocks of bitsize B on a server, in a binary tree structure of height $T = \lceil log_2 N \rceil$. Each node of the tree can store a constant amount Z of blocks. Every block encodes (in an encrypted form, using a semantically secure symmetric-key encryption scheme) a data unit of bitsize D, and optionally additional information which is used to label the block for efficient retrieval. There are many different ways one can implement this labeling of the blocks. In our case we will use the simple approach of concatenating to the data unit $\mathtt{data}$ a K-bit string encoding the block identifier $i \in \{1, \ldots, N\}$, that is, blocks are of the form $\mathtt{block}_i = \mathsf{Enc}_k(i \| \mathtt{data}_i)$. This system is very general, and as we will see it has the advantage that it easily translates to the quantum setting, unlike other approaches such as identifying blocks by using a hash table. At the beginning, all the blocks in the tree are initialized in an 'empty' state, which is defined by setting to 0 the identifying label - recall in fact that valid block identifiers are $1, \ldots, N$ only. Every block is mapped to a leaf of the tree, and this mapping is recorded in a so-called *position map* by the client[5]. A read (or write) operation for a block $\mathtt{block}_i$ is performed by the client, by downloading the path (tree branch) from the root of the tree to the leaf indicated in the client's position map, and $\mathtt{block}_i$ is randomly remapped to another leaf in the position map. Then the client decrypts and re-encrypts (re-randomizing) all the blocks in the downloaded path, and for every valid (non-empty) block $\mathtt{block}_j$ found, the client checks its corresponding leaf in the position map, and moves $\mathtt{block}_j$ (if there is enough available space) to the node in the path which is closest to the leaf level and that belongs both to the downloaded path and the path to the leaf of $\mathtt{block}_j$ given by the position map. If a block does not fit anywhere in the downloaded path, then an extra storage, called *'stash'* is used by the client to store this overflowing block locally. The blocks found in the stash are also examined during every read (or write) operation and checked if they can be evicted from the stash and placed in the tree. Since the stash must be stored locally, by the client, the stash's size should be reasonably small; in fact, in the paper's full version [41], the authors show that for any access pattern, the probability that the stash exceeds a size of $O(\log N)$ is negligible. The intuition is to notice that the stash is only used if the tree root is full, but the average action of a data request is to push only $\mathtt{block}_i$ toward the tree root, and push many other blocks $\mathtt{block}_j$ toward the leaf level. In the following we will ignore the use of the stash for simplicity.

More concretely, we give here a full description of PathORAM (which from now on we denote as $\mathtt{PathORAM}$) according to our new formalism.

[5] Note that the size of the position map is linear in the number of blocks that the client has, and thus cannot be stored locally by the client. The authors propose storing the position map recursively to smaller PathORAMs following an idea from [42]. For ease of exposition however (and without loss of generality), we will assume here that the position map is stored locally.

Definition 18 (PathORAM). *For fixed parameters $D, \widetilde{N} \in \mathbb{N}$, let $K = \lceil log_2 \widetilde{N} \rceil$, $Z \in \mathbb{N}$, $M = D + K$, $B \geq M$. Let $\mathcal{G}$ be a PRNG as from Definition 6 outputting K-bit values, and $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a SKE with M-bit plaintexts and B-bit ciphertexts. We define an ORAM construction called $\mathtt{PathORAM} = \mathtt{PathORAM}_{\mathcal{E},\mathcal{G}}$ as follows:*

- $\mathtt{PathORAM.Init}(n, N) \to (\mathcal{C}, \mathcal{S})$ *works in the following way:*
  1. *$\mathcal{C}$ generates a secret key $k \leftarrow \mathsf{KeyGen}(1^n)$;*
  2. *set $T = \lceil log_2 N \rceil$ (notice $T \leq K$);*
  3. *$\mathcal{C}$ initializes a lookup table (the* position map) of the form $((1, r_1), \ldots, (N, r_N))$, where r_i are T-bit values generated by truncating the first $K - T$ bits of $\mathcal{G}$'s output [6].
  4. $\mathcal{S}$.DB is stored in a binary tree of height T, with leaves $\mathtt{Leaf}_0, \ldots, \mathtt{Leaf}_{2^T-1}$, and such that:
     (a) each node of the tree stores up to Z blocks;
     (b) every block of every node is initialized to $\mathsf{Enc}_k(0^K \| 0^D)$.
- *If $\mathtt{dr} = (\mathtt{op}, i, \mathtt{data})$, then $\mathtt{PathORAM.Access}(\mathcal{C}, \mathcal{S}, \mathtt{dr}) \to (\mathcal{C}', \mathcal{S}', \mathtt{com})$ works in the following way:*
  1. *$\mathcal{S}$ sends to $\mathcal{C}$ the path $\mathtt{DPath}$ from the root of the tree to $\mathtt{Leaf}_{r_i}$;*
  2. *remap (i, r_i) to (i, r'_i) in the position map of $\mathcal{C}$, where r'_i is a fresh pseudorandom T-bit value (generated by truncating the first $K - T$ bits of $\mathcal{G}$'s output), obtaining $\mathcal{C}^*$;*
  3. *for every block $\mathtt{block}$ contained in $\mathtt{DPath}$, $\mathcal{C}^*$ does the following:*
     (a) *$\mathcal{C}^*$ decrypts $\mathsf{Dec}_k(\mathtt{block}) \to (j \| \mathtt{data}_j) \in \{0,1\}^M$, where, $j \in \{0,1\}^K$, $\mathtt{data}_j \in \{0,1\}^D$;*
     (b) *if $j = i$, then:*
        A. *if $\mathtt{op} =$ 'read', then $\mathcal{C}^*$ reads $\mathtt{data}_j$ ($\mathcal{C}^*$ is updated to $\mathcal{C}'$, which has* access *to $\mathtt{data}_j$);*
        B. *if $\mathtt{op} =$ 'write', then $\mathcal{C}$ sets $\mathtt{data}_j = \mathtt{data}$ (i.e., $\mathtt{block}$ is updated, so that $\mathtt{Data}(\mathtt{block}) = \mathtt{data}$), and set $\mathcal{C}' = \mathcal{C}^*$;*
     (c) *$\mathcal{C}'$ re-encrypts (re-randomizing) $\mathtt{block}$;*
     (d) *find in $\mathtt{DPath}$ the common parent node $\mathtt{Node}$ between $\mathtt{Leaf}_{r_i}$ and $\mathtt{Leaf}_{r_j}$, closer to the leaf level;*
     (e) *for every block $\mathtt{block}'$ in $\mathtt{Node}$, $\mathcal{C}'$ does the following:*
        A. *$\mathcal{C}'$ decrypts $\mathtt{block}'$, checks if it is an empty block, and re-encrypts (re-randomizing) it;*
        B. *if $\mathtt{block}'$ was found to be empty, then swap $\mathtt{block}$ and $\mathtt{block}'$;*
        C. *if no empty block was found in $\mathtt{Node}$, then set $\mathtt{Node}$ to be one level up in the tree (i.e., $\mathtt{Node}$'s parent) and repeat steps 3(e)A. to 3(e)C. (if $\mathtt{DPath}$ contains no empty block, use the stash);*
  4. *$\mathcal{C}'$ sends back the updated tree branch, $\mathtt{UPath}$, to $\mathcal{S}$;*
  5. *$\mathcal{S}$ updates $\mathcal{S}$.DB with $\mathtt{UPath}$, obtaining $\mathcal{S}'$;*
  6. *produce $\mathtt{com}$, which contains $r_i, \mathtt{DPath}, \mathtt{UPath}$.*

[6] It is easy to see that the truncated bits of $\mathcal{G}$ are also pseudorandom, since otherwise there would exist an algorithm that could distinguish between a random string and $\mathcal{G}$ output, by simply truncating the $K - T$ bits of both strings.

In the above, the meaning of the parameters is as follows:

- n is the security parameter, used by the encryption scheme $\mathcal{E}$.
- $\widetilde{N}$ is the maximum number of blocks that the server's architechture can support (an upper bound to $\mathcal{S}$'s tree storage).
- N is the maximum number of 'real' blocks that the client $\mathcal{C}$ wants to store (so, $N \leq \widetilde{N}$). Unlike $\widetilde{N}$ thus, N can be dictated by the adversary in the security game.
- K is the minimum number of bits that are needed to index all the 'real' blocks in the limit scenario where $N = \widetilde{N}$. Hence, K is also architecture-dependant, and not chosen by $\mathcal{A}$.
- Z is the maximum number of blocks that can be stored in every tree node. Lower values reduce the amount of memory used by $\mathcal{S}$ to store the tree (for a fixed N), but increase the risk of using large amounts of memory by the client for the stash. This is a parameter of the particular `PathORAM` implementation: as we do not care about performance here, we will leave Z undefined, as any nonzero value works for us in theory.
- T is the minimum number of bits that are needed to index all the 'real' N blocks (hence, $T \leq K$). T also represents the minimal height of the tree necessary to store all N blocks in the limit case where $Z = 1$.
- D is the bitsize of the data units used in the `PathORAM` implementation, and it is hence architecture-dependant.
- M is the total bitsize of a data unit, plus the number of bits necessary to address the block where this data unit is encoded, so also architecture-dependant. The encryption scheme $\mathcal{E}$ must be able to work with M-bit plaintexts.
- B is the size of a ciphertext produced by the encryption scheme $\mathcal{E}$, and hence the total size of a block. The size of $\mathcal{S}$'s tree storage memory is thus at most $B\widetilde{N}$ bits.

We are now ready to prove the (classical) security of `PathORAM` by using our new security framework.

Theorem 19. *Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be an IND-CPA SKE according to Definition 1, and let $\mathcal{G}$ be a PRNG as from Definition 6. Then, `PathORAM` instantiated using $\mathcal{E}$ and $\mathcal{G}$ is an AP-IND-CPA secure ORAM.*

Proof. By assumption, the outputs of $\mathcal{G}$ are indistinguishable from random. Therefore, in the following analysis, we can replace $\mathcal{G}$ with a real source of randomness, without loss of generality.

Suppose that there exists an adversary $\mathcal{A}$ and a non-negligible function ν, such that

$$\Pr\left[\mathsf{Game}^{\texttt{AP-IND-CQA}}_{\mathcal{A},\texttt{PathORAM}}(n) = 1\right] = \frac{1}{2} + \nu(n).$$

We will use $\mathcal{A}$ in a black-box way to construct a BPP algorithm $\mathcal{D}$, able to break the IND-CPA security of $\mathcal{E}$, against the assumption. The idea is to build an algorithm $\mathcal{D}$ which simulates a `PathORAM` client $\mathcal{C}$, playing the AP-IND-CQA game against $\mathcal{A}$ (without loss of generality, we assume that $\mathcal{A}$ itself simulates the server $\mathcal{S}$, otherwise $\mathcal{S}$ can be also simulated by $\mathcal{D}$). Throughout the game, $\mathcal{D}$ also stores a copy of the server's database $\mathcal{S}.\mathtt{DB}$, in plaintext. This is allowed, as $\mathcal{S}.\mathtt{DB}$ is of size linear in N, and $\mathcal{D}$ is only simulating $\mathcal{C}$, so she is not limited by the storage constraints usually assumed in a normal ORAM client. Then $\mathcal{D}$ will use the interaction with $\mathcal{A}$ to win the IND-CPA game for the scheme $\mathcal{E}$.

More in detail: $\mathcal{D}$ executes $\mathcal{A}$. $\mathcal{A}$ starts $\mathsf{Game}^{\mathtt{AP\text{-}IND\text{-}CQA}}_{\mathcal{A},\mathtt{PathORAM}}$ by choosing n and N, and $\mathcal{D}$ simulates a $\mathtt{PathORAM}$ client $\mathcal{C}$ created during $\mathtt{PathORAM.Init}$, by initializing his own position map (populated with random values), but *without* generating a secret encryption key. Furthermore, $\mathcal{D}$ creates a tree memory structure of height T, with leaves indexed $0, \ldots, 2^T - 1$, where every node stores Z plaintexts of bitsize M, which are initialized to $(0^K \| 0^D)$ (the parameters are the same as in Definition 18). This structure will be used by $\mathcal{D}$ to 'mirror' $\mathcal{S}.\mathtt{DB}$ in an unencrypted form throughout the execution of $\mathtt{PathORAM}$.

$\mathcal{D}$ now starts $\mathsf{Game}^{\mathtt{IND}}_{\mathcal{D}^{\mathsf{Enc}},\mathcal{E}}(n)$ according to Definition 1, obtaining oracle access to Enc_k for an unknown secret key k, and choosing as security parameter the same n chosen by $\mathcal{A}$. At this point, notice that $\mathcal{D}$ is able to perfectly simulate a valid client $\mathcal{C}$ having access to the key k, in the following way:

- whenever $\mathcal{C}$ would download a branch of $\mathcal{S}.\mathtt{DB}$ identified by leaf r, $\mathcal{D}$ does the same (although the blocks in such branch will be ignored, as we will see);
- whenever $\mathcal{C}$ would decrypt a certain block in a downloaded branch, $\mathcal{D}$ will simulate the decryption oracle Dec_k by fetching the plaintext $(i \| \mathtt{data})$ found at the corresponding position in the 'mirrored' tree;
- whenever $\mathcal{C}$ would swap two blocks in a downloaded branch, $\mathcal{D}$ will swap the two plaintexts found at the corresponding positions in the 'mirrored' tree;
- whenever $\mathcal{C}$ would encrypt a plaintext $(i \| \mathtt{data})$ to obtain a new encrypted block, $\mathcal{D}$ will do so by using the encryption oracle Enc_k from the IND-CPA game;
- whenever $\mathcal{C}$ would update his position map, or upload an updated branch to $\mathcal{S}.\mathtt{DB}$, $\mathcal{D}$ will do so.

Given the above, it is clear that now whenever $\mathcal{A}$ asks for the execution of a data request $\mathtt{dr}$, $\mathcal{D}$ is able to simulate the correct communication transcript $\mathtt{com}$ and a correctly formed updated branch $\mathtt{UPath}$. Therefore, for every data request during the first CQA phase, $\mathcal{A}$ always receives the correct access pattern.

Eventually, at the challenge query, $\mathcal{A}$ produces two data requests $\mathtt{dr}^0$ and $\mathtt{dr}^1$, and demands the execution of one of them. For $a \in \{0,1\}$, let $\mathtt{dr}^a = (\mathtt{op}^a, i^a, \mathtt{data}^a)$ be the two data requests and let $m^a \in \{0,1\}^M$ be formed as follows:

- if $\mathtt{op}^a =$ 'write', then set $m^a = (i^a \| \mathtt{data}^a)$;
- else, set $m^a = (i^a \| \mathtt{data}_{i^a})$, where $\mathtt{data}_{i^a}$ is retrieved by looking for identifier i^a in the mirrored tree.

Now, it could happen that $m^0 = m^1$. For example, it might be that the two data requests are of the form ('write', $i, \mathtt{data}$) and ('read', $i, \mathtt{data}'$) resp., but

$\texttt{block}_i$ already encodes $\texttt{data}$. If this happens we say that the challenge query is *non-meaningful*. It is easy to see that two data requests from a non-meaningful challenge query will produce the same statistical distributions of communication transcripts [7] and updated paths, because their effect on the database is equivalent. Therefore, since $\mathcal{A}$ distinguishes the two resulting access patterns with non-negligible probability by assumption, it is clear that the challenge query must be *meaningful*, i.e. $m^0 \neq m^1$.

At this point $\mathcal{D}$ executes the challenge IND query using m^0, m^1 as plaintexts, and receiving back an encryption $c \leftarrow \mathsf{Enc}_k(m^b)$ for a secret bit b. $\mathcal{D}$ will also generate a random bit $b^* \xleftarrow{\$} \{0,1\}$ (a 'guess'), and will answer $\mathcal{A}$'s challenge query by simulating the execution of $\texttt{dr}^{b^*}$ as in the CQA phase, but injecting c as an updated block with identifier i^{b^*} during the execution of $\texttt{dr}^{b^*}$. Then $\mathcal{D}$ keeps simulating $\mathcal{C}$ during the second CQA phase as before, and waits until $\mathcal{A}$ outputs a bit $\widehat{b}$. Finally: if $\widehat{b} = b^*$, then $\mathcal{D}$ outputs b^* in the IND-CPA game, otherwise $\mathcal{D}$ outputs a new random bit.

Now, notice the following. In the case that $\mathcal{D}$'s guess was correct, i.e. $b = b^*$, it means that c was the right ciphertext at the right place, so that $\mathcal{A}$ has received a correctly formed access pattern. This means that $\mathcal{A}$ correctly guesses $\widehat{b} = b^*$ with probability at least $\frac{1}{2} + \nu(n)$, by assumption. In that case, also $\mathcal{D}$ wins, so:

$$\Pr\left[\mathsf{Game}_{\mathcal{D},\mathcal{E}}^{\texttt{IND-CPA}}(n) = 1 \middle| b = b^*\right] \geq \frac{1}{2} + \nu. \tag{1}$$

On the other hand, if $b \neq b^*$ we cannot say anything on $\mathcal{A}$'s success probability, because now $\mathcal{A}$ has a malformed access pattern. But we can say that, even if $\mathcal{A}$ fails, $\mathcal{D}$ still succeeds with probability $\frac{1}{2}$.

$$\Pr\left[\mathsf{Game}_{\mathcal{D},\mathcal{E}}^{\texttt{IND-CPA}}(n) = 1 \middle| b \neq b^*\right] \geq \frac{1}{2}. \tag{2}$$

Thus, combining 1 and 2, the reduction's overall success probability is:

$$\Pr\left[\mathsf{Game}_{\mathcal{D},\mathcal{E}}^{\texttt{IND-CPA}}(n) = 1\right] \geq \frac{1}{2} + \frac{\nu}{2}$$

which concludes the proof. □

We stress how this is the first full formal proof of $\texttt{PathORAM}$'s security, its simplicity made possible by the new security model we introduced.

4 Post-Quantum ORAMs

In this section we study the post-quantum security of classical ORAMs. First of all, we show that the extension of a classically secure ORAM to its post-quantum secure counterpart is not trivial. To this end, we examine $\texttt{PathORAM}$ and we show

[7] Notice how this is not true anymore if the values in the position map are not totally random. Therefore, this step fails if the PRNG used is not secure.

that merely substituting the underlying encryption scheme with a post-quantum secure one does not generally yield a pq-AP-IND-CQA ORAM. The idea is to exploit the weakness of other components of the ORAM construction under examination (in this case, the PRNG used). This is mostly a theoretical result, since it has to be somewhat expected that post-quantum security can only be achieved by hardening all the underlying components, not only the encryption scheme.

Then, we show that building post-quantum secure ORAMs is possible. We do it in a black-box way by showing that `PathORAM`, instantiated with a post-quantum secure SKE *and* a post-quantum secure PRNG, achieves post-quantum security. This is important from an application perspective, because it shows that efficient and post-quantum secure ORAMs can indeed be obtained in a straightforward way.

4.1 The Impossibility Result

In order to show that one cannot in general obtain post-quantum secure ORAMs by just using a post-quantum SKE in a black-box way, we provide the following counterexample.

Theorem 20. *Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a pq-IND-CPA SKE according to Definition 2, and let $\mathcal{G}^*$ be the PRNG from Lemma 7. Let $\mathtt{PathORAM}^*$ be the ORAM obtained by instantiating the $\mathtt{PathORAM}$ construction from Definition 18 using $\mathcal{E}$ and $\mathcal{G}^*$, i.e., $\mathtt{PathORAM}^* = \mathtt{PathORAM}_{\mathcal{E},\mathcal{G}^*}$. Then, under the DLP hardness assumption, $\mathtt{PathORAM}^*$ is AP-IND-CQA secure, but not pq-AP-IND-CQA secure.*

At the light of Theorem 19 and Definition 15, in order to prove Theorem 20 we only need the following lemma.

Lemma 21. *There exist a QPT algorithm $\mathcal{A}$ and a non-negligible function ν such that $\mathcal{A}$ wins the game $\mathsf{Game}^{\mathit{AP\text{-}IND\text{-}CQA}}_{\mathcal{A},\mathtt{PathORAM}^*}(n)$ with probability at least $\frac{1}{2} + \nu(n)$.*

Proof. We start by making a key observation concerning the access patterns produced in `PathORAM`. Let $\mathtt{dr} = (\mathtt{op}, i, \mathtt{data})$ be a data request sent by $\mathcal{C}$. By only examining the communication transcript `com` resulting from the execution of this data request, one can see which path (branch of the tree) $\mathcal{S}$ sent to $\mathcal{C}$, thus learning the leaf r_i to which i was mapped to, even without knowing i itself. In normal circumstances, this is of no use to an adversary, because this value r_i becomes immediately obsolete, being replaced by a new fresh value output by the PRNG in the position map. But it will be important in our attack as we will see.

Let $\mathcal{D}$ be the *BQP* algorithm (the 'PRNG predictor') of Lemma 7. We build the adversary $\mathcal{A}$ with oracle access to $\mathcal{D}$. First of all, $\mathcal{A}$ chooses $n, N \leq \widetilde{N}$ and starts the AP-IND-CQA game by calling $\mathtt{PathORAM}^*.\mathtt{Init}(n, N)$. For his attack, $\mathcal{A}$ fixes an arbitrary identifier $i \in \{1, \ldots, N\}$, and an arbitrary data unit $\mathtt{data} \in \{0,1\}^D$.

During the first CQA learning phase, $\mathcal{A}$ asks $\mathcal{C}$ to execute $k = \text{poly}(n)$ consecutive data requests of the form ('write', i, $\texttt{data}$). $\mathcal{A}$ records the resulting access patterns from all these queries, $\texttt{ap}_1, \ldots, \texttt{ap}_k$, which include the communication transcripts $\texttt{com}_1, \ldots, \texttt{com}_k$ and then, by the observation made before, a 'history' $(r_i^{(0)}, \ldots, r_i^{(k-1)})$ of the past mappings of block i at the beginning of the execution of every data request from 1 to k. These mappings, in turn, are k outputs of $\mathcal{PRG}^*$, and they are given as input to the algorithm $\mathcal{D}$, which then outputs a candidate prediction r^* for the current secret leaf value $r_i^{(k)}$.

Then $\mathcal{A}$ executes his challenge query by using data requests $(\texttt{dr}^0, \texttt{dr}^1)$ with $\texttt{dr}^0 =$ ('write', i, $\texttt{data}$), and $\texttt{dr}^1 =$ ('write', j, $\texttt{data}$) for $j \neq i$, and records the resulting access pattern $\texttt{ap}_{k+1} = \texttt{ap}(\texttt{dr}^b)$ (where b is the secret bit to be guessed). At this point, the adversary looks at this last communication transcript $\texttt{com}_{k+1}$ and, by the observation made at the beginning of the proof, checks the leaf index r related to the tree branch exchanged during the execution of the challenge query. If $r = r^*$, then $\mathcal{A}$ sets $b' = 0$ (where b' is $\mathcal{A}$'s current 'guess' at b), otherwise $\mathcal{A}$ sets $b' = 1$.

However, before outputting his guess b' in order to win the AP-IND-CQA game, $\mathcal{A}$ has to perform an additional check (during the second CQA challenge phase) in order to verify whether $\mathcal{D}$ had correctly guessed the right value $r_i^{(k)}$ or not. The problem here is that, if $\mathcal{D}$ is unsuccessful (which happens with probability as high as $1 - \delta$), we cannot say anything about the predicted value r^*. In fact, in that case $\mathcal{D}$ could potentially act maliciously against $\mathcal{A}$, and output a value r^* which maximizes the probability of b' being wrong in the above strategy: for example, $r^* = r_j^{(0)}$. For this reason $\mathcal{A}$ performs the following 'sanity check' after the challenge query:

- if $b' = 1$, then $\mathcal{A}$ demands the execution of an additional query of the form ('write', i, $\texttt{data}$), and verifies that the resulting path leads to leaf r^*. This guarantees that r^* was actually correct, and it was not observed during the challenge query just because $\texttt{dr}^1$ was chosen, as guessed.
- Otherwise, if $b' = 0$, then $\mathcal{A}$ demands the execution of an additional query of the form ('write', j, $\texttt{data}$), and verifies that the resulting tree branch *does not* lead to leaf r^*. This guarantees with high probability that $\mathcal{D}$ did not maliciously output the secret leaf state for element j instead of i.

It is easy to see that in the case of misbehavior of $\mathcal{D}$, both of the above tests fail with high probability. In fact, in the case $b' = 1$, the current mapping of element i leads to leaf $r_i^{(k)}$, which was *not* correctly predicted by $\mathcal{D}$ by assumption. In the latter case instead, recall that $\mathcal{A}$ had guessed $b' = 0$ because during the execution of the challenge query he observed the leaf r^*; this could only lead to a fail in the case that $r_j^{(0)} = r_i^{(k)}$, which only happens with negligible probability at most ϵ, or if $r_j^{(0)} = r^*$, which is detected by the sanity check.

Finally, if the above sanity check is passed, $\mathcal{A}$ outputs b', otherwise he outputs a random bit.

Notice that (provided $\mathcal{D}$ was successful) this strategy is always correct, *except* in the case that: $\texttt{dr}_1$ was chosen (probability $\frac{1}{2}$) *and* the initial mapping of $\texttt{block}_j$

(which is $r_j^{(0)}$), coincides with $r_i^{(k)}$. As already mentioned, the latter event can only happen at most with probability ϵ negligible in the bit size of $\mathcal{PRG}^*$'s output, and hence in the security parameter n (it is easy to see that this is a minimum requirement for any classically secure PRNG, as $\mathcal{PRG}^*$ is). Thus:

$$\Pr\left[\mathsf{Game}_{\mathcal{A},\mathtt{PathORAM}^*}^{\mathtt{AP\text{-}IND\text{-}CQA}} \to 1 \middle| \mathcal{D} \text{ succeeds}\right] \geq 1 - \frac{\epsilon}{2}. \tag{3}$$

On the other hand, if $\mathcal{D}$ fails (which happens with probability $(1-\delta)$ at most) and predicts a wrong value $r^* \neq r_i^{(k)}$, the above strategy still succeeds with probability at least $\frac{1}{2} - \frac{\epsilon}{2}$ (again, because of the remote possibility that $r_j^{(0)} = r_i^{(k)}$). Hence:

$$\Pr\left[\mathsf{Game}_{\mathcal{A},\mathtt{PathORAM}^*}^{\mathtt{AP\text{-}IND\text{-}CQA}} \to 0 \middle| \mathcal{D} \text{ fails}\right] \leq \frac{1}{2}(1+\epsilon). \tag{4}$$

Thus, combining 3 and 4, the adversary's overall success probability is:

$$\Pr\left[\mathsf{Game}_{\mathcal{A},\mathtt{PathORAM}^*}^{\mathtt{AP\text{-}IND\text{-}CQA}} \to 1\right]$$

$$= \Pr[\mathcal{A} \text{ wins}] \cdot \Pr[\mathcal{D} \text{ succeeds}] + (1 - \Pr[\mathcal{A} \text{ loses}] \cdot \Pr[\mathcal{D} \text{ fails}])$$
$$\geq \delta\left(1 - \frac{\epsilon}{2}\right) + \left(1 - (1-\delta)\frac{1}{2}(1+\epsilon)\right) \geq \frac{1}{2} + \frac{1}{2}\delta - \frac{1}{2}\epsilon,$$

which concludes the proof, because ϵ is negligible, while δ is not. □

Theorem 20 in particular shows a separation between classical and post-quantum security for ORAMs.

Corollary 22. *Under the classical DLP hardness assumption, there exist ORAMs which are AP-IND-CQA secure but not pq-AP-IND-CQA secure.*

4.2 Construction of a Post-Quantum ORAM

A careful examination of `PathORAM`'s construction details reveals that an important role in the security is played by the pseudorandom number generator used to map a block to a leaf during every access. As we have just shown, a PRNG which is not post-quantum secure is enough to break `PathORAM`'s security in a quantum setting. It is natural then to wonder whether the attack on `PathORAM` can be avoided by using a post-quantum secure PRNG, *in addition* to a post-quantum secure encryption scheme, when instantiating `PathORAM`. In this section, we give a positive answer to such question.

Theorem 23. *Let $\mathcal{E}$ be a pq-IND-CPA SKE according to Definition 1, and let $\mathcal{G}$ be a pq-PRNG as from Definition 6. Then, `PathORAM` instantiated using $\mathcal{E}$ and $\mathcal{G}$ is a pq-AP-IND-CPA secure ORAM.*

Proof. The proof follows step-by-step the proof of Theorem 19. In fact this time, since $\mathcal{G}$ is a pq-PRNG by assumption, the new output values used to update the position map in `PathORAM` are indistinguishable from random (and therefore, in particular, unpredictable) even for QPT adversaries. As $\mathcal{G}$ has an internal state which is completely unrelated to $\mathcal{E}$'s internal randomness, the security arguments at every step in the proof of Theorem 19 remain unchanged. Therefore, any QPT adversary who can distinguish the execution of two data request sequences with probability non-negligibly better than guessing, can be turned into a successful adversary against the pq-IND-CPA security of $\mathcal{E}$, or against the pq-PRNG, against the security assumptions. $\square$

5 Quantum ORAM

In this section we initiate the study of *quantum ORAMs (QORAM)*, that is, ORAM constructions operating on *quantum data*. This new cryptographic primitive that we define considers the same scenario as in the ORAM case, but where all the parties have quantum computing and communication capabilities. In this respect, QORAM are not an example of post-quantum security, but belong to the more general area of *quantum security* [14], as in the case of QKD [6] or quantum bit commitments [44].

In our QORAM model, the client $\mathcal{C}$ and the server $\mathcal{S}$ are both BQP algorithms, sharing a *quantum communication channel* (quantum register) Ψ. Since such a quantum channel can also be used to share classical information, we assume without loss of generality that $\mathcal{A}$ and $\mathcal{S}$ also share a classical channel Ξ. In the following, if not otherwise stated, we will always assume that all the classical communication between $\mathcal{A}$ and $\mathcal{S}$ happens through Ξ, and all the quantum communication happens through Ψ. In this scenario, a computationally limited $\mathcal{C}$ wants to outsource a *quantum database (QDB)* to the more powerful $\mathcal{S}$, and perform operations on the QDB in a secure way, as in the ORAM case.

We have first to define what it means to have a 'quantum database'. In our case, this will be a structure of *quantum blocks*. A quantum block is a B-qubit quantum state $\psi \in \mathfrak{D}(\mathcal{H}_B)$ for a fixed parameter $B \in \mathbb{N}$ which depends on $\mathcal{C}$'s and $\mathcal{S}$'s architectures. A *quantum database* (QDB) of size $N \in \mathbb{N}$ is a quantum register of $\mathcal{S}$ which stores N quantum blocks. It is important to notice that we impose no restriction on the nature of the states stored in the quantum blocks, i.e., these states could be mixed or entangled, amongst them or with states stored in other, external registers. But in the following, for simplicity, we abuse notation and denote such multipartite system with a tuple of quantum blocks $(\psi_1, \ldots, \psi_N)$. Since we assume this quantum register to reside on the server's side, we will denote it as $\mathcal{S}.\mathtt{QDB}$. As in the ORAM case, the precise way this system of quantum blocks is represented in the quantum database is unspecified, and left to the exact implementation of the QORAM scheme taken into account. As usual, we will abuse notation and write that $\mathcal{S}.\mathtt{QDB}(i) = \psi$ if ψ is the state obtained by tracing out all but the i-th subsystem of $\mathcal{S}.\mathtt{QDB}$, and that $\psi \in \mathcal{S}.\mathtt{QDB}$ if $\mathcal{S}.\mathtt{QDB}(i) = \psi$ for some $i \in \mathbb{N}$.

A quantum block encodes (usually in an encrypted form) a *quantum data unit*, which is another quantum state representing the information that the client actually wants to access or modify, and possibly additional (quantum or classical) auxiliary information. Formally, a quantum data unit is a quantum state $\varphi \in \mathfrak{D}(\mathcal{H}_D)$ of D qubits, where $D \leq B$ depends on $\mathcal{C}$'s and $\mathcal{S}$'s architecture. As before, no assumption is made about the nature of these quantum states. Every quantum block can encode a single quantum data unit, therefore at any given time t it is defined a function $\mathtt{QData}_t : \mathcal{S}.\mathsf{QDB} \to \mathfrak{D}(\mathcal{H}_D)$. With abuse of notation, we will denote by $\mathtt{QData}(\psi)$ the quantum data unit encoded in the block ψ at a certain time. The client $\mathcal{C}$ can operate on the quantum database through *quantum data requests*.

Definition 24 (Quantum Data Request)**.** *A* quantum data request *to a database $\mathcal{S}.\mathsf{QDB}$ is a tuple of the form $\mathtt{qdr} = (\mathtt{op}, i, \varphi)$, where $\mathtt{op} \in \{read, write\}, i \in \{1, \ldots, N\}$, and $\varphi \in \mathfrak{D}(\mathcal{H}_D)$ is a quantum data unit.*

Finally, we define the meaning of a *quantum communication transcript* during an execution of a QORAM protocol. As in the ORAM case, we will use the following definition.

Definition 25 (Quantum Communication Transcript)**.** *A* quantum communication transcript *$\mathtt{qcom}$ at time t is the content of the communication registers (Ξ, Ψ) at time t of the protocol's execution.*

As in the ORAM case, in the following we will consider $\mathtt{qcom}$ as a discrete function of the round $1, 2, \ldots$ of the protocol. Notice the following difference from the classical case: as this time $\mathcal{C}$ and $\mathcal{S}$ are also allowed to exchange quantum data through Ψ, it might not be possible for an adversary to obtain a full transcript of $\mathtt{qcom}$ without disturbing the protocol. We will address this issue in the next section about security.

From now on, B and D will be fixed constants (the *quantum block size*, and *quantum data unit size*, resp.) As in the classical case, we assume that a server's QDB is always initialized empty (that is, with randomized encryptions of $|0 \ldots 0\rangle$ elements as data), and it is left up to the client the task of 'populating' the database. We are now ready to define a QORAM as follows.

Definition 26 (QORAM)**.** *Let $\widetilde{N} \in \mathbb{N}, M \geq D$, and $\mathcal{E}_Q = (Q.\mathsf{KeyGen}, Q.\mathsf{Enc}, Q.\mathsf{Dec})$ be a QSKE scheme according to Definition 3, mapping M-qubit plaintext states to B-qubit ciphertext states. A* QORAM *$\mathtt{QORAM}_{\mathcal{E}_Q}$ with parameters $(D, \widetilde{N}, \mathcal{E}_Q)$ is a pair of two-party interactive quantum algorithms, which we denote by $(\mathtt{QORAM.Init}, \mathtt{QORAM.Access})$, such that:*

- *$\mathtt{QORAM.Init}(n, N) \to (\mathcal{C}, \mathcal{S})$ in the following way:*
  1. *n is the security parameter, $N < \widetilde{N}$;*
  2. *$k \leftarrow Q.\mathsf{KeyGen}(1^n)$ is generated by $\mathcal{C}$;*
  3. *$\mathcal{S}$ includes a QDB $\mathcal{S}.\mathsf{QDB} = (\psi_1, \ldots, \psi_N)$.*
- *$\mathtt{QORAM.Access}(\mathcal{C}, \mathcal{S}, \mathtt{qdr}) \to (\mathcal{C}', \mathcal{S}', \mathtt{qcom})$ in the following way:*
  1. *$\mathcal{C}$ issues a quantum data request $\mathtt{qdr}$;*

2. $\mathcal{C}$ and $\mathcal{S}$ communicate via (Ξ, Ψ) and produce the quantum communication transcript `qcom`.

The same considerations about soundness hold as in the classical case, with the following exception. In the case of quantum data we say that an element $\varphi \in \mathfrak{D}(\mathcal{H})$ is *accessible* by a quantum circuit M if M can simulate *one single invocation* of the quantum oracle $\mathcal{O}_\varphi(|0\dots0\rangle\langle0\dots0|) \mapsto \varphi$, where $|0\dots0\rangle\langle0\dots0| \in \mathfrak{D}(\mathcal{H})$. This limitation is needed because requiring unlimited access to the oracle can be used to clone the given state, which is forbidden by the *No-Cloning Theorem*. Multiple calls to $\mathcal{O}_\varphi$ can be allowed for certain states, i.e., if M himself knows how to build φ from scratch, but in the general case it is sufficient to require one single call to such oracle in order to define *accessibility* of a quantum state. The same considerations for the efficiency of the oracle simulation hold here as in the classical case.

5.1 QORAM Security

We now look at the security model for QORAMs. As in the classical model, security will be given in terms of adaptive access pattern indistinguishability.

Our threat model considers a quantum adversary $\mathcal{A}$, which we identify as $\mathcal{S}$ himself, and who wants to compromise $\mathcal{C}$'s privacy by having access to the communication channel (Ξ, Ψ) and $\mathcal{S}$'s internal memory, but who is not allowed to modify the content of the channel against the protocol. Without loss of generality, we assume that the only meaningful changes in respect to $\mathcal{A}$ only happen in the database area $\mathcal{S}$.`QDB` between the beginning and the end of a `QORAM.Access` execution.

As it often happens in the quantum world, there is a caveat here: it is unclear what a 'honest-but-curious' quantum adversary is. In fact, the problem is even more general: we do not have a notion of 'read-only' for quantum channels, as the mere act of observing the data in transit through Ψ can destroy such data.

For example, suppose that a quantum state φ is sent through Ψ. Because of the No-Cloning Theorem, $\mathcal{S}$ cannot store a local copy of φ; at the same time, measuring φ in transit through Ψ without any knowledge of such state, would disturb it with high probability. Therefore, it seems hard to justify the inclusion of the state φ in the adversarial view (the *quantum access pattern*) in the quantum model for a honest-but-curious adversary.

Nevertheless, we think it is important to allow the adversary $\mathcal{A}$ to know some information about the quantum state φ. There are many reasons for this choice. First of all, remember that we are defining QORAMs in a very abstract and general way, and the exact details of how the communication and storage of quantum information works is left to the particular QORAM construction. For example, there might be constructions which only use quantum states from a finite, fixed set of orthogonal states, or which only use subsets of quantum states admitting efficient classical representations (and encoding them in a classical way during the communication). Moreover, as the adversary $\mathcal{A}$ is usually allowed to choose the initial state of the quantum database, as in the classical model, it might be possible that he has access to some side-information which allows him to know something about the content of the database or the data transferred in a sound way, e.g. by applying some quantum operation or partial measurement which does not disturb the state too much. As we need to cover all these possibilities, we find the option of not including the quantum data in the access pattern too restrictive. On the other hand, the adversary $\mathcal{A}$ should not be able to modify too much (from $\mathcal{C}$'s point of view) any quantum state, as this would go beyond the notion of honest-but-curious adversary usually considered in the ORAM scenario.

We will solve this issue by introducing a *safe extractor*. The intuition behind this novel technique is to allow our adversary to extract any kind of (quantum) information he wants from a certain physical system, *as long as such extraction is hardly noticeable by any other party*. In this case we say that the action of the adversary on the physical system is *computationally undetectable*.

Definition 27 (Safe Extractor)**.** *Let φ_A be the state contained in a quantum register A. A* safe extractor *for A in the state φ_A is a BQP algorithm χ with additional classical input x of size polynomial in n, acting on A and outputting a quantum state ψ of qubit size polynomial in n, and such that the action of χ on φ_A is computationally undetectable.*

Here, *computationally undetectable* means that no QPT algorithm can reliably distinguish whether a quantum operation takes place or not by just looking at the processed quantum state, even in presence of auxiliary information such as, e.g., additional entangled registers. More formally:

Definition 28 (Computational Undetectability of Quantum Action)**.** *Let L, A, B be quantum registers of size polynomial in n, and φ_A a quantum state on A. A quantum algorithm Λ acting on L and A has* computationally undetectable action on φ_A *iff for any bipartite quantum state φ_{AB} such that $(\varphi_{AB})_A = \varphi_A$, and for any BQP algorithm $\mathcal{D}$ acting on A and B and outputting 0 or 1:*

$$\left|\Pr\left[\mathcal{D}\left(\varphi_{AB}\right)=1\right]-\Pr\left[\mathcal{D}\left(\left(\Lambda\otimes\mathbb{I}_B\right)\left(|0\rangle\langle 0|_L\otimes\varphi_{AB}\right)_{AB}\right)=1\right]\right|\leq negl(n).$$

Notice that Definition 27 depends on the state contained in the quantum register considered. That is, χ might be a safe extractor for a given quantum register if that register is in a certain state, but not in a different one. Of course one could define χ to be a safe extractor for a register *tout court* if it is a safe extractor for *any* state of that register according to Definition 27, but this would considerably reduce the power of the adversary. Instead, this definition allows the adversary to use χ adaptively, only at certain points of his execution, when he is guaranteed that the action of χ on the current state of the QORAM will be computationally undetectable. The additional classic input to χ serves a useful purpose here, as it can be seen as a way for the adversary to communicate instructions to χ about how to perform the extraction in a safe way (for example, $\mathcal{A}$ might encode a certain measurement basis through this classical input.) With abuse of notation, and without loss of generality, we will write $\psi \leftarrow \chi(\mathtt{qcom}, \mathcal{S}.\mathtt{QDB})$ to denote that χ performs the following:

– as a classical input, χ gets the classical part of a quantum communication transcript $\mathtt{qcom}$ (that is, the content of the classical channel Ξ) and additional classical information by $\mathcal{A}$;
– χ acts on the quantum registers Ψ and $\mathcal{S}.\mathtt{QDB}$;
– finally, χ produces a quantum output ψ.

More specifically, we define a QORAM adversary as follows.

Definition 29 (QORAM Adversary)**.** *A* QORAM adversary *is a BQP algorithm $\mathcal{A}^\chi$ with quantum oracle access to χ, where:*

– *$\mathcal{A}$ has complete control of $\mathcal{S}$, as long as the QORAM's soundness is preserved;*
– *χ is a* safe extractor *for the joint register $(\mathcal{S}.\mathtt{QDB}, \Psi)$ for any of its states at any moment during the execution of $\mathcal{A}$.*

We stress the fact that our safe extractor technique can be generalized to many other scenarios. In fact, it expresses in a general way the intuition behind a plethora of techniques which have been independently used in many other works, see for example [10, 12, 11, 3]. Although specific applications might need a refinement of the definition, we believe this new technique to be a very general tool of independent interest, which can be useful in the study of different quantum security reductions.

We are now able to define *quantum access patterns*, as the outputs of the safe extractor before and after the execution of a quantum data request.

Definition 30 (Quantum Access Pattern)**.** *Given a QORAM client and server $\mathcal{C}$ and $\mathcal{S}$, a quantum data request $\mathtt{qdr}$, and a QORAM adversary $\mathcal{A} = \mathcal{A}^\chi$, the* quantum access pattern observed by $\mathcal{A}$*, denoted by $\mathtt{qap}_{\mathcal{A}}(\mathtt{qdr})$, is the pair of quantum states (ψ, ψ'), where:*

– *$\psi \leftarrow \chi(\mathtt{qcom}, \mathcal{S}.\mathtt{QDB})$;*
– *$(\mathcal{C}', \mathcal{S}', \mathtt{qcom}') \leftarrow \mathtt{QORAM.Access}(\mathcal{C}, \mathcal{S}, \mathtt{qdr})$*
– *$\psi' \leftarrow \chi(\mathtt{qcom}', \mathcal{S}'.\mathtt{QDB})$.*

Notice that, since the action of the safe extractor is computationally undetectable, running it on two consecutive quantum data requests does not allow, in any case, to clone unknown quantum states. We define the new security game as follows.

Definition 31 ($\mathsf{Game}^{\mathtt{QAP\text{-}IND\text{-}CQA}}_{\mathcal{A},\mathtt{QORAM}}(n)$)**.** *Let $\mathtt{QORAM} = (\mathtt{QORAM.Init}, \mathtt{QORAM.Access})$ be a QORAM construction, n a security parameter and $\mathcal{A} = \mathcal{A}^\chi$ a QORAM adverary. The* computational indistinguishability of quantum access patterns game under adaptive chosen query attack $\mathsf{Game}^{\mathtt{QAP\text{-}IND\text{-}CQA}}_{\mathcal{A},\mathtt{QORAM}}(n)$ *proceeds as follows:*

1. *$\mathcal{A}$ chooses $N \leq \widetilde{N}$;*
2. *$(\mathcal{C}, \mathcal{S}) \leftarrow \mathtt{QORAM.Init}(n, N)$;*
3. *first CQA learning phase: for $i = 1, \ldots, q_1 \in \mathbb{N}$, $\mathcal{A}$ repeats (adaptively) the following:*

(a) $\mathcal{A}$ *chooses a quantum data request* $\texttt{qdr}_i$*;*
(b) $\mathcal{C}$ *executes* $\texttt{QORAM.Access}$ *on* $\texttt{qdr}_i$*;*
(c) $\mathcal{A}$ *receives* $\texttt{qap}_{\mathcal{A}}(\texttt{qdr}_i)$*;*

4. *challenge phase:* $\mathcal{A}$ *chooses two quantum data requests* $\texttt{qdr}^0$ *and* $\texttt{qdr}^1$*;*
5. $\mathcal{C}$ *flips a random secret bit* $b \xleftarrow{\$} \{0,1\}$ *and executes* $\texttt{QORAM.Access}$ *on* $\texttt{qdr}^b$*;*
6. $\mathcal{A}$ *receives* $\texttt{qap}_{\mathcal{A}}(\texttt{qdr}^b)$*;*
7. *second CQA learning phase: for* $j = 1, \ldots, q_2 \in \mathbb{N}$*,* $\mathcal{A}$ *repeats (adaptively) the following:*
   (a) $\mathcal{A}$ *chooses a quantum data request* $\texttt{qdr}_j$*;*
   (b) $\mathcal{C}$ *executes* $\texttt{QORAM.Access}$ *on* $\texttt{qdr}_j$*;*
   (c) $\mathcal{A}$ *receives* $\texttt{qap}_{\mathcal{A}}(\texttt{qdr}_j)$*;*
8. $\mathcal{A}$ *outputs a bit* b'*.*

$\mathcal{A}$ *wins the game iff* $b = b'$

We are now ready to define the security notion that we will use for QORAMs.

Definition 32 (Quantum Access Pattern Indistinguishability Under Adaptive Chosen Query Attack)**.** *A QORAM construction* $\texttt{QORAM}$ *has* computationally indistinguishable quantum access patterns under adaptive chosen query attack *(or, it is QAP-IND-CQA-secure) iff for any QORAM adversary* $\mathcal{A}$*:*

$$\left| \Pr\left[\mathsf{Game}^{\texttt{QAP-IND-CQA}}_{\mathcal{A},\texttt{QORAM}}(n) = 1 \right] - \frac{1}{2} \right| \leq negl(n).$$

5.2 `PathQORAM`

In this section we describe the construction for a novel QAP-IND-CQA-secure QORAM scheme, which we call *`PathQORAM`*, and which has the interesting property that read and write operations are inherently equivalent. The idea is to modify PathORAM with the quantum symmetric-key encryption scheme proposed in [1], but we need some additional care for ensuring soundness. In fact, we have the following problem. Suppose the client issues a quantum data request for block i. This will be translated to a leaf in $\mathcal{S}$'s quantum database, and the resulting tree branch `DPath` will be sent to $\mathcal{C}$. Now $\mathcal{C}$ knows that the data he is looking for is encoded in one of `DPath`'s nodes, but he does not know which one. Classically, $\mathcal{C}$ would proceed by decrypting and inspecting every node in `DPath` until he finds what he is looking for, then he would perform some operation on that element, before re-encrypting it again, and then complete the re-randomization of `DPath` before re-sending the whole branch to $\mathcal{S}$. This operation might be problematic in the quantum world though: inspecting an unknown quantum state will destroy it with high probability. We have therefore to find a way to signal $\mathcal{C}$ when he reaches the right node in the path without disturbing the quantum data unit itself. The solution is to notice that, in our formalization of PathORAM, the client stores the classical identifier i together with the data unit in the block. In the quantum version `PathQORAM`, this identifier is still classical, and of a fixed length K. Once a node in `DPath` is decrypted, it will be transformed to $|i\rangle\langle i|\otimes\varphi$. The first register can then be measured in the computational basis without being disturbed, and without disturbing the state φ (which is not entangled with $|i\rangle$). So the trick for $\mathcal{C}$ is to find out when he is decrypting the right element by *only* measuring the first K qubits of the decrypted block, and only act on the quantum data unit when the right identifier is found. Notice how other different approaches used classically to instantiate PathORAM, such as identifying blocks by storing a local table with the hash values of the data units, might not work so smoothly when translated to the quantum world.

More concretely, we give here a full description of PathQORAM (which from now on we denote as `PathQORAM`) according to our new formalism. The meaning of the parameters is as in Definition 18.

Definition 33 (`PathQORAM`). *For fixed parameters $D, \widetilde{N} \in \mathbb{N}$, let $K = \lceil log_2 \widetilde{N} \rceil$, $Z \in \mathbb{N}$, $M = D+K$, $B \geq M$. Let $\mathcal{G}$ be a pqPRNG as from Definition 6 outputting K-bit values, and $\mathcal{E}_Q = (Q.\mathsf{KeyGen}, Q.\mathsf{Enc}, Q.\mathsf{Dec})$ be a QSKE with plaintext space $\mathfrak{D}(\mathcal{H}_M)$ and ciphertext space $\mathfrak{D}(\mathcal{H}_B)$. We define a QORAM construction called $\mathtt{PathQORAM} = \mathtt{PathQORAM}_{\mathcal{E}_Q,\mathcal{G}}$ as follows:*

- *$\mathtt{PathQORAM.Init}(n, N) \to (\mathcal{C}, \mathcal{S})$ works in the following way:*
  1. *$\mathcal{C}$ generates a secret key $k \leftarrow Q.\mathsf{KeyGen}(1^n)$;*
  2. *set $T = \lceil log_2 N \rceil$ (notice $T \leq K$);*
  3. *$\mathcal{C}$ initializes a lookup table (the* position map) of the form $((1, r_1), \ldots, (N, r_N))$, where r_i are T-bit values generated by truncating the first $K-T$ bits of $\mathcal{G}$'s output[8];
  4. $\mathcal{S}.\mathtt{QDB}$ is stored in a binary tree of height T, with leaves $\mathtt{Leaf}_0, \ldots, \mathtt{Leaf}_{2^T-1}$, and such that:
     (a) each node of the tree stores up to Z quantum blocks;
     (b) every quantum block of every node is initialized to $Q.\mathsf{Enc}_k(|0^M\rangle\langle 0^M|)$.
- *If $\mathtt{qdr} = (\mathtt{op}, i, \varphi)$, then $\mathtt{PathQORAM.Access}(\mathcal{C}, \mathcal{S}, \mathtt{qdr}) \to (\mathcal{C}', \mathcal{S}', \mathtt{qcom})$ works in the following way:*
  1. *$\mathcal{S}$ sends to $\mathcal{C}$ the path $\mathtt{DPath}$ from the root of the tree to $\mathtt{Leaf}_{r_i}$;*
  2. *remap (i, r_i) to (i, r'_i) in the position map of $\mathcal{C}$, where r'_i is a fresh value output by $\mathcal{G}$, obtaining $\mathcal{C}^*$;*
  3. *for every quantum block ψ contained in $\mathtt{DPath}$, $\mathcal{C}^*$ does the following:*
     (a) $\mathcal{C}^*$ decrypts $Q.\mathsf{Dec}_k(\psi) \to |j\rangle\langle j| \otimes \sigma$, where $|j\rangle \in \mathcal{H}_K$, and $\sigma \in \mathfrak{D}(\mathcal{H}_D)$;
     (b) $\mathcal{C}^*$ measures the first K qubits of the decrypted state in the computational basis, obtaining j;
     (c) if $j = i$, then swap σ with φ ($\mathcal{C}^*$ is updated to $\mathcal{C}'$, which has access *to σ, and at the same time the block is updated so that $\mathtt{QData}(\psi) = \varphi$);*
     (d) $\mathcal{C}'$ re-encrypts (re-randomizing) ψ;
     (e) find in $\mathtt{DPath}$ the common parent node $\mathtt{Node}$ between $\mathtt{Leaf}_{r_i}$ and $\mathtt{Leaf}_{r_j}$, closer to the leaf level;
     (f) for every quantum block ψ' in $\mathtt{Node}$, $\mathcal{C}'$ does the following:

[8] the pseudorandomness of these T bits follows from the same argument as in the classical case, described in Def. 18

A. $\mathcal{C}'$ decrypts ψ', checks if it is an empty block (the first K qubits decrypt to $|0^K\rangle\langle 0^K|$), and re-encrypts (re-randomizing) it;

B. if ψ' was found to be empty, then swap ψ and ψ';

C. if no empty quantum block was found in `Node`, then set `Node` to be one level up in the tree (i.e., `Node`'s parent) and repeat steps 3(f)A. to 3(f)C. (if `DPath` contains no empty quantum block, use the stash);

4. $\mathcal{C}'$ sends back the updated tree branch, `UPath`, to $\mathcal{S}$;

5. $\mathcal{S}$ updates $\mathcal{S}$.`QDB` with `UPath`, obtaining $\mathcal{S}'$;

6. produce `qcom`, which contains r_i, `DPath`, `UPath`.

Notice that the following interesting property holds: the operations of 'write' and 'read' have the *same* effect. Namely: since qubits from the server's database cannot be copied, and cannot be removed or added (otherwise this would compromise indistinguishability), the action of a read or write operation is simply to swap a state in the database with a state in $\mathcal{C}$'s memory. In fact, `QORAM.Access` swaps φ known by $\mathcal{C}$ with σ stored in $\mathcal{S}$. Also notice how `qcom` containing `DPath`, `UPath` would imply a cloning of quantum states. This is just a formal artifice, because in the case of QORAMs as we defined them, `qcom` is only used in respect to a safe extractor χ, which will only produce classical information. For the soundness of the `PathQORAM` construction we have left unexplained the use of a *quantum stash*. This is an area of quantum memory basically used as the classical stash of `PathORAM`, but every time an element is 'written' in the stash, it is actually 'swapped' with an empty block in the tree. The security of the construction follows from the Q-IND-CPA security of the quantum encryption scheme $\mathcal{E}_Q$, and from the security of the pqPRNG $\mathcal{G}$.

Theorem 34. *Let $\mathcal{E}_Q$ be a Q-IND-CPA SKE according to Definition 5, and let $\mathcal{G}$ be a pq-PRNG as from Definition 6. Then, `PathQORAM` instantiated using $\mathcal{E}_Q$ and $\mathcal{G}$ is a QAP-IND-CPA secure QORAM.*

The proof can be found in Appendix A.3: it basically follows the steps of the proof of Theorem 23, but with an additional tweak. In fact, the reduction $\mathcal{D}$ cannot store a local unencrypted copy of the tree, but she can store a tree of the (classical) unencrypted identifiers with the same mapping of $\mathcal{S}$.`QDB` at any time frame. When simulating $\mathcal{C}$ for a given quantum data request, $\mathcal{D}$ can hence identify all the blocks in the downloaded path, but cannot re-randomize them. She will work around this issue by replacing every time the encrypted blocks with 'artificial' blocks obtained by encrypting (through the encryption oracle of $\mathcal{E}_Q$) the 'right' identifier and a $|0\dots0\rangle$ data unit. The success probability of the adversary $\mathcal{A}$ cannot be affected too much by this substitution, otherwise $\mathcal{D}$ could use this fact to break the Q-IND-CPA of $\mathcal{E}_Q$.

Acknowledgments We are grateful to the anonymous reviewers for insightful comments, and to Marc Fischlin and Christian Schaffner for many fruitful discussions. This work has been funded by CYSEC, CRISP, and the DFG as part of projects S4 and S5 within the CRC 1119 CROSSING.

References

1. Gorjan Alagic, Anne Broadbent, Bill Fefferman, Tommaso Gagliardoni, Christian Schaffner, and Michael St. Jules. Computational security of quantum encryption. In *ICITS*, 2016.
2. Andris Ambainis, Ansis Rosmanis, and Dominique Unruh. Quantum attacks on classical proof systems: The hardness of quantum rewinding. In *FOCS*, 2014.
3. Mayuresh Vivekanand Anand, Ehsan Ebrahimi Targhi, Gelo Noel Tabia, and Dominique Unruh. Post-quantum security of the CBC, CFB, OFB, CTR, and XTS modes of operation. In *PQCrypto*, 2016.
4. Marco Baldi, Marco Bianchi, Franco Chiaraluce, Joachim Rosenthal, and Davide Schipani. Enhanced public key security for the mceliece cryptosystem. *J. Cryptology*, 2016.
5. Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 394–403, 1997.
6. Charles H. Bennett and Gilles Brassard. An update on quantum cryptography. In *CRYPTO*, 1984.
7. Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 1978.
8. Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen. *Post-Quantum Cryptography*. Springer-Verlag, 2009.
9. Erik-Oliver Blass, Travis Mayberry, Guevara Noubir, and Kaan Onarlioglu. Toward robust hidden volumes using write-only oblivious RAM. In *CCS*, 2014.
10. Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random Oracles in a Quantum World. In *ASIACRYPT 2011*, 2011.
11. Dan Boneh and Mark Zhandry. Quantum-secure message authentication codes. In *EUROCRYPT*, 2013.
12. Dan Boneh and Mark Zhandry. Secure signatures and chosen ciphertext security in a quantum computing world. In *CRYPTO*, 2013.
13. Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *STOC*, 2013.
14. Anne Broadbent and Christian Schaffner. Quantum cryptography beyond quantum key distribution. *Des. Codes Cryptography*, 2016.
15. Ivan Damgård, Jakob Funder, Jesper Buus Nielsen, and Louis Salvail. Superposition attacks on cryptographic protocols. In *ICITS*, 2013.
16. Tommaso Gagliardoni, Andreas Hülsing, and Christian Schaffner. Semantic security and indistinguishability in the quantum world. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, pages 60–89, 2016.
17. Sanjam Garg, Payman Mohassel, and Charalampos Papamanthou. TWORAM: efficient oblivious RAM in two rounds with applications to searchable encryption. In *CRYPTO 2016*, 2016.
18. Craig Gentry, Shai Halevi, Charanjit Jutla, and Mariana Raykova. Private database access with HE-over-ORAM architecture. IACR ePrint, 2014/345, 2014.
19. Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

20. Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious RAMs. *J. ACM*, 1996.
21. Michael T. Goodrich, Michael Mitzenmacher, Olga Ohrimenko, and Roberto Tamassia. Practical oblivious storage. In *CODASPY*, 2012.
22. Michael T. Goodrich, Michael Mitzenmacher, Olga Ohrimenko, and Roberto Tamassia. Privacy-preserving group data access via stateless oblivious RAM simulation. In *SODA*, 2012.
23. Lov K. Grover. A fast quantum mechanical algorithm for database search. In *ACM Symposium on the Theory of Computing*, 1996.
24. Elloá B. Guedes, Francisco Marcos de Assis, and Bernardo Lula Jr. Quantum attacks on pseudorandom generators. *Mathematical Structures in Computer Science*, 2013.
25. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *ANTS-III*, 1998.
26. Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. Breaking symmetric cryptosystems using quantum period finding. *CoRR*, 2016.
27. Elham Kashefi and Iordanis Kerenidis. Statistical zero knowledge and quantum one-way functions. *Theor. Comput. Sci.*, 2007.
28. Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 2007.
29. Eyal Kushilevitz, Steve Lu, and Rafail Ostrovsky. On the (in)security of hash-based oblivious RAM and a new balancing scheme. In *SODA*, 2012.
30. Hidenori Kuwakado and Masakatu Morii. Quantum distinguisher between the 3-round feistel cipher and the random permutation. In *ISIT 2010*, 2010.
31. Hidenori Kuwakado and Masakatu Morii. Security on the quantum-type even-mansour cipher. In *ISITA 2012*, 2012.
32. Vadim Lyubashevsky and Daniele Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In *CRYPTO*, 2009.
33. Martin Maas, Eric Love, Emil Stefanov, Mohit Tiwari, Elaine Shi, Krste Asanovic, John Kubiatowicz, and Dawn Song. PHANTOM: practical oblivious computation in a secure processor. In *CCS*, 2013.
34. Matteo Maffei, Giulio Malavolta, Manuel Reinert, and Dominique Schröder. Privacy and access control for outsourced personal records. In *IEEE S&P*, 2015.
35. Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In *CRYPTO*, 2013.
36. Cristopher Moore, Alexander Russell, and Umesh Vazirani. A classical one-way function to confound quantum adversaries. quant-ph/0701115, 2007.
37. Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
38. Benny Pinkas and Tzachy Reinman. Oblivious RAM revisited. In *CRYPTO*, 2010.
39. Elaine Shi, T.-H. Hubert Chan, Emil Stefanov, and Mingfei Li. Oblivious RAM with o((logn)3) worst-case cost. In *ASIACRYPT*, 2011.
40. Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *FOCS*, 1994.
41. Emil Stefanov and Elaine Shi. Path O-RAM: an extremely simple oblivious RAM protocol. *CoRR*, abs/1202.5150, 2012.
42. Emil Stefanov, Elaine Shi, and Dawn Xiaodong Song. Towards practical oblivious RAM. In *NDSS*, 2012.
43. Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher W. Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM: an extremely simple oblivious RAM protocol. In *CCS*, 2013.

44. Dominique Unruh. Computationally binding quantum commitments. *IACR ePrint*, 2015.
45. Xiao Shaun Wang, Yan Huang, T-H. Hubert Chan, abhi shelat, and Elaine Shi. SCORAM: Oblivious RAM for secure computation. IACR ePrint, 2014/671, 2014.
46. Peter Williams, Radu Sion, and Bogdan Carbunar. Building castles out of mud: practical access pattern privacy and correctness on untrusted storage. In *CCS*, 2008.
47. Peter Williams, Radu Sion, and Alin Tomescu. Privatefs: a parallel oblivious file system. In *CCS 2012*, 2012.
48. Mark Zhandry. How to construct quantum random functions. In *FOCS*, 2012.

A Appendix: Additional Proofs

A.1 Proof of Lemma 7

Proof. The counterexample $\mathcal{G}^*$ we use in this proof is the modular exponentiation Blum-Micali generator [28], but many other similar variants work as well [24]. This construction is based on exponentiation of a public generator g modulo a public large prime p, and it is a classically secure PRNG under the assumption that computing discrete logarithms is computationally hard. More specifically, if s_i is the current state of the generator, one output bit is computed as a hardcore predicate $h(.)$ of the value $s_{i+1} = g^{s_i} \mod p$ (where s_{i+1} becomes the next state of the generator). Thus, starting from a single secret seed s_0, a pseudorandom bitstring can be generated by applying iteratively the procedure.

However, there exists a quantum attack [24] (based on variants of both Shor's [40] and Grover's [23] algorithms) which, given p, g and a sequence $(r_1, \ldots, r_n)$ of values output by $\mathcal{G}^*$, can recover the initial state s_0 with non-negligible probability. This, in turns, allows to predict the whole sequence of outputs of $\mathcal{G}^*$. $\square$

A.2 Proof of Theorem 17

Proof. The proof closely follows the reduction for equivalent notions of IND-CPA security found in [5]. As a first step, we rephrase Definition 16 in terms of a security game – the two resulting definitions of AP-SIM-CQA security are obviously equivalent.

Definition 35 ($\mathsf{Game}^{\texttt{AP-SIM-CQA}}_{\mathcal{A},\texttt{ORAM}}(n)$)**.** *Let* $\texttt{ORAM} = (\texttt{ORAM.Init}, \texttt{ORAM.Access})$ *be an ORAM construction, n a security parameter and $\mathcal{A}$ an ORAM adverary. The* computational simulability of access patterns game under adaptive chosen query attack $\mathsf{Game}^{\texttt{AP-SIM-CQA}}_{\mathcal{A},\texttt{ORAM}}(n)$ *proceeds as follows:*

1. *$\mathcal{A}$ chooses $N \leq \widetilde{N}$;*
2. *$(\mathcal{C}, \mathcal{S}) \leftarrow \texttt{ORAM.Init}(n, N)$;*
3. *$\mathcal{C}$ flips a secret random bit $b \xleftarrow{\$} \{0,1\}$;*
4. *$\mathcal{A}$ repeats (adaptively) the following, for $i = 1, \ldots, q$:*
   (a) $\mathcal{A}$ chooses a data request $\texttt{dr}_i$;

(b) *if* $b = 0$*, then* $\mathcal{C}$ *sets* $\mathtt{dr} := \mathtt{dr}_i$*;*

(c) *else if* $b = 1$*, then* $\mathcal{C}$ *sets* $\mathtt{dr} \xleftarrow{\$} \{ \text{'read'}, \text{'write'} \} \times \{ 1, \ldots, N \} \times \{0,1\}^D$*;*

(d) $\mathcal{C}$ *executes* $\mathtt{ORAM.Access}$ *on* $\mathtt{dr}$*;*

(e) $\mathcal{A}$ *receives* $\mathtt{ap}(\mathtt{dr}_i)$*;*

Finally, $\mathcal{A}$ *outputs a bit* b'*, and wins the game iff* $b = b'$*.*

Definition 36 ((Game-Based) Access Pattern Simulability Under Adaptive Chosen Query Attack)**.** *An ORAM construction* $\mathtt{ORAM}$ *has* simulable access patterns under adaptive chosen query attack *(or, it is AP-SIM-CQA-secure) iff for any classical ORAM adversary* $\mathcal{A}$*:*

$$\left| \Pr\left[\mathsf{Game}^{\mathtt{AP\text{-}SIM\text{-}CQA}}_{\mathcal{A},\mathtt{ORAM}}(n) = 1 \right] - \frac{1}{2} \right| \leq negl(n).$$

Then, we prove Theorem 17 by double implication. Let $\mathtt{ORAM}$ be an ORAM construction.

Lemma 37 (AP-SIM-CQA $\implies$ AP-IND-CQA)**.**

Proof of Lemma 37. Let $\mathcal{A}^{\mathtt{IND}}$ be an ORAM adversary such that

$$\Pr\left[\mathsf{Game}^{\mathtt{AP\text{-}IND\text{-}CQA}}_{\mathcal{A}^{\mathtt{IND}},\mathtt{ORAM}}(n) = 1 \right] = \frac{1}{2} + \nu(n),$$

for a non-negligible function ν. We will use $\mathcal{A}^{\mathtt{IND}}$ in a black-box way to construct another ORAM adversary $\mathcal{A}^{\mathtt{SIM}}$, able to break the AP-SIM-CQA security of $\mathtt{ORAM}$, against the assumption.

At the beginning, $\mathcal{A}^{\mathtt{SIM}}$ runs $\mathcal{A}^{\mathtt{IND}}$, which starts $\mathsf{Game}^{\mathtt{AP\text{-}IND\text{-}CQA}}_{\mathcal{A}^{\mathtt{IND}},\mathtt{ORAM}}$ by choosing n and N. At the same time, $\mathcal{A}^{\mathtt{SIM}}$ starts $\mathsf{Game}^{\mathtt{AP\text{-}SIM\text{-}CQA}}_{\mathcal{A}^{\mathtt{SIM}},\mathtt{ORAM}}$ with the same parameters.

During the first IND learning phase, whenever $\mathcal{A}^{\mathtt{IND}}$ performs a CQA query $\mathtt{dr}_i$, $\mathcal{A}^{\mathtt{SIM}}$ responds to such a query by forwarding it to its SIM (Real-or-Random) challenger, and then forwarding to $\mathcal{A}^{\mathtt{IND}}$ the related access pattern. When $\mathcal{A}^{\mathtt{IND}}$ performs its challenge query of the form $(\mathtt{dr}^0, \mathtt{dr}^1)$, $\mathcal{A}^{\mathtt{SIM}}$ flips a random bit $b^* \xleftarrow{\$} \{0,1\}$, forwards $\mathtt{ap}^{b^*}$ to the SIM challenger, and then forwards to $\mathcal{A}^{\mathtt{IND}}$ the related access pattern. Then, during the second IND learning phase, $\mathcal{A}^{\mathtt{SIM}}$ behaves like in the first challenge phase, by forwarding the queries to its SIM challenger and returning the access patterns to $\mathcal{A}^{\mathtt{IND}}$. Finally, when $\mathcal{A}^{\mathtt{IND}}$ outputs a bit b', if $b' = b^*$ then $\mathcal{A}^{\mathtt{SIM}}$ outputs 0, otherwise $\mathcal{A}^{\mathtt{SIM}}$ outputs a random bit.

The reduction works for the following reason: let's assume that $b = 0$, i.e., the SIM challenger was a honest ORAM client $\mathcal{C}$. Then, during the whole reduction, $\mathcal{A}^{\mathtt{SIM}}$ successfully simulated a real client for $\mathcal{A}^{\mathtt{IND}}$, in an AP-IND-CQA game where the secret bit was b^*. By assumption, $\mathcal{A}^{\mathtt{IND}}$ guesses correctly such bit with probability at least $\frac{1}{2} + \nu(n)$. In that case, also $\mathcal{A}^{\mathtt{SIM}}$ wins, so:

$$\Pr\left[\mathsf{Game}^{\mathtt{AP\text{-}SIM\text{-}CQA}}_{\mathcal{A}^{\mathtt{SIM}},\mathtt{ORAM}}(n) = 1 \middle| b = 0 \right] \geq \frac{1}{2} + \nu. \tag{5}$$

On the other hand, if $b = 1$ (i.e., the SIM challenger was simulating fake access patterns) we cannot say anything on $\mathcal{A}^{\texttt{IND}}$'s success probability, because for the whole time it has played in a malformed game. But we can say that, even if $\mathcal{A}^{\texttt{IND}}$ fails, $\mathcal{A}^{\texttt{SIM}}$ still succeeds with probability $\geq \frac{1}{2} - \varepsilon$ for a negligible ε.

$$\Pr\left[\mathsf{Game}^{\texttt{AP-SIM-CQA}}_{\mathcal{A}^{\texttt{SIM}},\texttt{ORAM}}(n) = 1 \middle| b = 1\right] \geq \frac{1}{2} - \varepsilon. \tag{6}$$

Thus, combining 5 and 6, the reduction's overall success probability is:

$$\Pr\left[\mathsf{Game}^{\texttt{AP-SIM-CQA}}_{\mathcal{A}^{\texttt{SIM}},\texttt{ORAM}}(n) = 1\right] \geq \frac{1}{2} + \frac{\nu}{2} - \varepsilon,$$

which concludes the proof. □

Lemma 38 (AP-IND-CQA $\implies$ AP-SIM-CQA)**.**

Proof of Lemma 38. Let $\mathcal{A}^{\texttt{SIM}}$ be an ORAM adversary such that

$$\Pr\left[\mathsf{Game}^{\texttt{AP-SIM-CQA}}_{\mathcal{A}^{\texttt{SIM}},\texttt{ORAM}}(n) = 1\right] \geq \frac{1}{2} + \nu(n),$$

for a non-negligible function ν. We will use $\mathcal{A}^{\texttt{SIM}}$ in a black-box way to construct another ORAM adversary $\mathcal{A}^{\texttt{IND}}$, able to break the AP-IND-CQA security of ORAM, against the assumption. The proof uses a hybrid argument on the number q of queries performed by $\mathcal{A}^{\texttt{SIM}}$ during $\mathsf{Game}^{\texttt{AP-SIM-CQA}}_{\mathcal{A}^{\texttt{SIM}},\texttt{ORAM}}$.

More in detail: $\mathcal{A}^{\texttt{IND}}$ runs $\mathcal{A}^{\texttt{SIM}}$, which starts $\mathsf{Game}^{\texttt{AP-SIM-CQA}}_{\mathcal{A}^{\texttt{SIM}},\texttt{ORAM}}$ by choosing n and N. At the same time, $\mathcal{A}^{\texttt{IND}}$ starts $\mathsf{Game}^{\texttt{AP-IND-CQA}}_{\mathcal{A}^{\texttt{IND}},\texttt{ORAM}}$ with the same parameters, and also chooses an index $j \xleftarrow{\$} \{1, \ldots, q\}$ uniformly at random. Whenever $\mathcal{A}^{\texttt{SIM}}$ performs a query with a data request $\texttt{dr}_i$, $\mathcal{A}^{\texttt{IND}}$ does the following:

- for the first $j-1$ queries, $\mathcal{A}^{\texttt{IND}}$ executes $\texttt{dr}_i$ using his own oracle in the first CQA learning phase, and responds with $\texttt{ap}(\texttt{dr}_i)$.
- At the j-th query, $\mathcal{A}^{\texttt{IND}}$ does the following:
  - samples at random $\texttt{dr}^* \xleftarrow{\$} \{\text{`read'}, \text{`write'}\} \times \{1, \ldots, N\} \times \{0,1\}^D$;
  - set $\texttt{dr}^0 := \texttt{dr}_j$ and $\texttt{dr}^1 := \texttt{dr}^*$.
  - performs his AP-IND-CQA challenge query $(\texttt{dr}_0, \texttt{dr}_1)$, and responds to $\mathcal{A}^{\texttt{SIM}}$ with $\texttt{ap}(\texttt{dr}^b)$.
- Starting from the $(j+1)$-th query, $\mathcal{A}^{\texttt{IND}}$ ignores $\mathcal{A}^{\texttt{SIM}}$'s data requests, and always responds with access patterns produced by freshly generated random data requests using his own oracle in the second CQA learning phase.

Finally, when $\mathcal{A}^{\texttt{SIM}}$ outputs a bit b', if $b' = 0$ then $\mathcal{A}^{\texttt{IND}}$ outputs 0, otherwise $\mathcal{A}^{\texttt{IND}}$ outputs a random bit. We show that:

$$\Pr\left[\mathsf{Game}^{\texttt{AP-IND-CQA}}_{\mathcal{A}^{\texttt{IND}},\texttt{ORAM}}(n) = 1\right] \geq \frac{1}{2} + \frac{\nu}{2q}.$$

The hybrid argument goes as follows. For $i = 0, \ldots, q$ consider a sequence of intermediate hybrid games $\mathsf{Game}^{\texttt{HYB-i}}_{\mathcal{A}^{\texttt{SIM}},\texttt{ORAM}}$, where $\mathsf{Game}^{\texttt{HYB-i}}_{\mathcal{A}^{\texttt{SIM}},\texttt{ORAM}}$ is defined as the usual $\mathsf{Game}^{\texttt{AP-SIM-CQA}}_{\mathcal{A}^{\texttt{SIM}},\texttt{ORAM}}$ by executing through the SIM (Real-or-Random) oracle only the first i data requests output by $\mathcal{A}^{\texttt{SIM}}$, and executing random data requests afterwards. Notice that:

- $\mathsf{Game}^{\texttt{HYB-q}}_{\mathcal{A}^{\texttt{SIM}},\texttt{ORAM}}$ coincides with $\mathsf{Game}^{\texttt{AP-SIM-CQA}}_{\mathcal{A}^{\texttt{SIM}},\texttt{ORAM}}$, and
- $\mathsf{Game}^{\texttt{HYB-0}}_{\mathcal{A}^{\texttt{SIM}},\texttt{ORAM}}$ coincides with $\mathsf{Game}^{\texttt{AP-SIM-CQA}}_{\mathcal{A}^{\texttt{SIM}},\texttt{ORAM}}$ in the case that $b = 1$.

Moreover, the following hold:

- the output b' of $\mathcal{A}^{\texttt{SIM}}$ in the above reduction coincides with the output of $\mathcal{A}^{\texttt{SIM}}$ in $\mathsf{Game}^{\texttt{HYB-(i+1)}}_{\mathcal{A}^{\texttt{SIM}},\texttt{ORAM}}$ if $b = 0$, while
- the output b' of $\mathcal{A}^{\texttt{SIM}}$ in the above reduction coincides with the output of $\mathcal{A}^{\texttt{SIM}}$ in $\mathsf{Game}^{\texttt{HYB-i}}_{\mathcal{A}^{\texttt{SIM}},\texttt{ORAM}}$ if $b = 1$.

At this point, as j was chosen uniformly at random in $\{1, \dots, q\}$, we can write the advantage of $\mathcal{A}^{\texttt{IND}}$ in $\mathsf{Game}^{\texttt{AP-IND-CQA}}_{\mathcal{A}^{\texttt{IND}},\texttt{ORAM}}(n)$ by use of the triangular inequality as follows:

$$\Pr\left[\mathcal{A}^{\texttt{IND}} \text{ outputs } 1 \middle| b = 1\right] - \Pr\left[\mathcal{A}^{\texttt{IND}} \text{ outputs } 1 \middle| b = 0\right] =$$

$$\frac{1}{q}\sum_{i=0}^{q-1}\left(\Pr\left[\mathcal{A}^{\texttt{SIM}} \text{ outputs 1 in } \mathsf{Game}^{\texttt{HYB-i}}_{\mathcal{A}^{\texttt{SIM}},\texttt{ORAM}}\right] - \Pr\left[\mathcal{A}^{\texttt{SIM}} \text{ outputs 1 in } \mathsf{Game}^{\texttt{HYB-(i+1)}}_{\mathcal{A}^{\texttt{SIM}},\texttt{ORAM}}\right]\right) =$$

$$\frac{1}{q}\left(\Pr\left[\mathcal{A}^{\texttt{SIM}} \text{ outputs 1 in } \mathsf{Game}^{\texttt{HYB-0}}_{\mathcal{A}^{\texttt{SIM}},\texttt{ORAM}}\right] - \Pr\left[\mathcal{A}^{\texttt{SIM}} \text{ outputs 1 in } \mathsf{Game}^{\texttt{HYB-q}}_{\mathcal{A}^{\texttt{SIM}},\texttt{ORAM}}\right]\right) =$$

$$\frac{1}{q}\left(\Pr\left[\mathcal{A}^{\texttt{SIM}} \text{ outputs 1 in } \mathsf{Game}^{\texttt{HYB-0}}_{\mathcal{A}^{\texttt{SIM}},\texttt{ORAM}} \middle| b = 1\right] - \Pr\left[\mathcal{A}^{\texttt{SIM}} \text{ outputs 1 in } \mathsf{Game}^{\texttt{HYB-q}}_{\mathcal{A}^{\texttt{SIM}},\texttt{ORAM}}\right]\right) \geq$$

$$\frac{1}{2q}\left[\text{ advantage of } \mathcal{A}^{\texttt{SIM}} \text{ in } \mathsf{Game}^{\texttt{AP-SIM-CQA}}_{\mathcal{A}^{\texttt{SIM}},\texttt{ORAM}}\right] = \frac{\nu}{2q}$$

which concludes the proof. □

The combination of Lemma 37 and Lemma 38 concludes the proof.

□

A.3 Proof of Theorem 34

Proof. The proof follows step-by-step the proof of Theorem 19 with some important differences. First of all, $\mathcal{D}$ cannot store a local mirrored tree of plaintexts of the form $(|i\rangle\langle i| \otimes \sigma)$ because of the No-Cloning Theorem, so she cannot simulate $\mathcal{C}$ perfectly. But she can store a mirrored tree which contains *only* the classical identifiers i, at the right positions of every block throughout the execution of the protocol.

At this point, $\mathcal{D}$ can simulate a decryption oracle for a certain block ψ in a downloaded branch by fetching the cleartext identifier i found at the corresponding position in the 'mirrored' tree, and creating a 'simulated' plaintext of the form $(|i\rangle\langle i| \otimes |0^D\rangle\langle 0^D|)$, i.e. replacing the 'real' quantum data unit σ

with a zero state. Since $\mathcal{A}$ never 'sees' a decrypted block, this substitution is not immediately apparent to him. Moreover, whenever $\mathcal{C}$ would create a block by encrypting $\psi \leftarrow Q.\mathsf{Enc}_k(|i\rangle\langle i| \otimes \sigma)$, $\mathcal{D}$ can simulate this by doing $\psi \leftarrow Q.\mathsf{Enc}_k(|i\rangle\langle i| \otimes |0^D\rangle\langle 0^D|)$. By the Q-IND-CPA security of $\mathcal{E}_Q$, $\mathcal{A}$ cannot detect this substitution with probability more than negligible. Therefore, $\mathcal{D}$ can still simulate $\mathcal{C}$ (with overwhelming, albeit not 100%, probability) at any data request.

Another issue appears during the challenge phase, as this time the concept of *non-meaningful* challenge must be redefined. For the same argument as above, from $\mathcal{A}$'s perspective it does not matter whether two data requests lead to two 'different' quantum data units σ^0, σ^1 (the analogue of data units $\mathtt{data}^0, \mathtt{data}^1$ in the classical proof) or not. Therefore, $\mathcal{D}$ can ignore the quantum data units at all. Moreover, as discussed in Section 5.2, in `PathQORAM` there is no difference between 'read' and 'write' operations. It follows, from the same argument as in the proof of Theorem 19, that the two challenge quantum data requests $\mathtt{qdr}^0, \mathtt{qdr}^1$ must differ on the identifiers i^0, i^1. $\mathcal{D}$, will then play the Q-IND-CPA game with challenge plaintexts $\varphi^a = |i^a\rangle\langle i^a| \otimes |0^D\rangle\langle 0^D|$ for $a \in \{0,1\}$, following the same strategy as in the classical case (by guessing a bit, injecting the challenge ciphertext, and observing $\mathcal{A}$'s output) , with only a negligible loss in the success probability because she is simulating fake plaintexts. $\square$