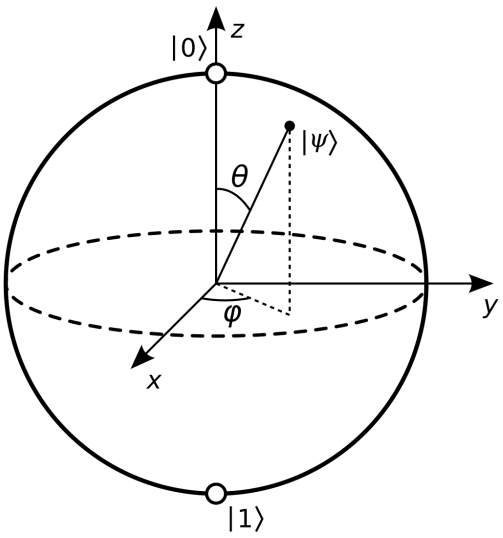
Post-quantum zero-knowledge with depth-efficient quantum simulation

Research work under the supervision of Alex B. Grilo & Damien Vergnaud

Gaspard Damoiseau-Malraux
Sorbonne Université

Feb. 10, 2025 — Aug. 1, 2025



Academic supervisor: Mohab Safey El Din

SORBONNE UNIVERSITÉ

LIP6

1 Introduction

Cryptology (or the **science of secrecy**) is a discipline of mathematics and computer science which includes both the design of encryption systems and secure protocols (cryptography), and the study of these systems and the evaluation of their security (cryptanalysis). Today, cryptology is the science that enables the implementation of confidential communications, from banking technologies to instant messaging applications, and more generally, the entirety of the internet.

Zero-knowledge proofs (ZKPs), introduced in 1985 by Goldwasser et al. [GMR85], are central to modern cryptology. Initially presented as theoretical tools, they now find concrete applications in authentication systems and, more recently, in blockchain systems.

A zero-knowledge proof is an interactive protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$, at the end of which the prover must have convinced the verifier of a statement, without the latter having learned anything in the process. Informally, this absence of information is often defined as follows: "what $\mathcal{V}$ sees in the protocol (even if he cheats) should be something which [he] could have computed himself" (Goldwasser et al., [GMR89]). By this definition, we see that the *zero-knowledge* nature of such a protocol largely depends on the computational capabilities of the verifier, which is defined as a polynomial runtime algorithm, in λ the security parameter.

In order to prove the zero-knowledge nature of a protocol, it is generally required to exhibit a simulator, whose output should be statistically indistinguishable from the transcript of a real (and usually successful) execution of the protocol. The simulator should have the same time/memory complexities as the verifier — the underlying idea is that the simulator, devoid of the secret, cannot reveal any information to which the verifier would not already have access. Therefore, if the simulator is capable of producing a transcript that is indistinguishable from a genuine execution, it means that simply executing the protocol, even with dishonest inputs, does not let the adversary learn any new information.

In the classical model, the scientific community knows how to produce these simulators in a relatively simple manner. But as quantum technologies develop, cryptographers are forced to rethink the requirements of their cryptographic tools, to achieve a level of post-quantum security. The threat model then shifts from a classical polynomial time/memory model to a quantum polynomial time/memory model. Currently, although the scientific community knows many post-quantum ZK protocols, all of them require a simulator with a nonnegligible complexity overhead. As such, there is currently an ongoing effort towards improving the relative complexity of simulators for post-quantum zero-knowledge.

Thus, when considering the situation where the simulator has approximately the same quantum memory as the verifier (up to a polynomial p, i.e., where $\text{Qubits}(\mathsf{Sim}) = p(\text{Qubits}(\mathcal{V}))$, we find a hard limit on the amount of quantum memory to which $\mathcal{V}$ must have access. It has been proven by Ananth and Grilo [AG22] that any language admitting a black-box zero-knowledge protocol, where the verifier has a super-logarithmic

quantum memory, necessarily belongs in **BQP**.[1] Put differently, this means that secure post-quantum zero-knowledge with space-bounded simulations cannot exist when the verifier has access to a super-logarithmic amount of qubits (in λ).

It is in the continuation of this work that my M2 internship takes place: studying the depth of simulators for post-quantum zero-knowledge. The depth of a circuit is a quantum analogue of the classical time complexity, and corresponds to the number of "slices" of a quantum circuit, or seen differently, the amount of "steps" that some qubits go through before measurement (even though these steps can have parallel operations). The open question we are working on is the following:

What kind of languages admit zero-knowledge protocols, where there exist depth-efficient (i.e., constant-depth) simulators?

The goal of our research is to prove that only languages in **BQP** admit simulators whose depth remains constant — that is, there exist no depth-efficient simulators for nontrivial languages. For example, if we assume that a language $\mathbb{L} \notin \mathbf{BQP}$, the contrapositive of this conjecture would imply that there cannot exist a ZKP with a constant-depth simulator for $\mathbb{L}$. This kind of results would contribute to shape the theoretical field of post-quantum ZK; in the classical setting, Goldreich et al. proved that, assuming the existence of one-way functions, all languages in **NP** admit zero-knowledge proofs [GMW91]. However, not all are secure: easy languages (for example, ones in **P**) also admit ZKPs, but none of them offer the security that is desired from a ZKP, because a polynomial adversary has the computing ability to find their own witness. Similarly, proving that the existence of a constant-depth verifier implies membership to **BQP** would help characterize and differentiate what is possible, and what is useful.

In addition — and perhaps the most practical consequence —, it would also help to know what can be expected from existing quantum devices. One main difficulty is indeed the lifespan of qubits, because it is currently hard for experimentalists to keep qubits coherent for more than a few milliseconds. The impossibility of a constant-depth circuit would therefore hinder near future implementations.

In the work we present below, we lay the first steps towards proving this research goal. In particular, we show that for any zero-knowledge protocol which admits an efficient simulator, we can always build another simulator that requires only a constant number of effective queries to $\mathcal{V}$. In addition, we propose a strategy and intuition for the next stages of the proof, suggesting that a simulator with a constant number of effective queries to $\mathcal{V}$ would imply that the language associated to such a post-quantum ZKP is in **BQP**.

[1] **BQP** (*Bounded-error Quantum Polynomial time*) is a complexity class that contains probabilistic quantum algorithms with a polynomial execution time, a quantum equivalent to complexity class **P**. Similarly, **QMA** (*Quantum Merlin-Arthur*) is a quantum equivalent to **NP**.

2 Zero-knowledge

In this section, we will present the concept of zero-knowledge proofs in more detail. A zero-knowledge protocol is an interactive proof system between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$, in which $\mathcal{P}$ is to prove that a statement is true without revealing anything more than the veracity of this statement. The concept of ZK interactive proofs was introduced in 1985 by Goldwasser, Micali, and Rackoff [GMR85]. An interactive proof system for a language $\mathbb{L}$ is a valid zero-knowledge protocol if it verifies three fundamental properties:

1. **Completeness:** at the end of the execution, the verifier must accept an honest prover with high probability if $\mathcal{P}$'s witness $x \in \mathbb{L}$ (i.e., "the protocol must work").
2. **Soundness:** the probability that the verifier accepts a cheating prover must be negligible if $x \notin \mathbb{L}$ (i.e., "$\mathcal{V}$ should not accept $\mathcal{P}$ is the statement is false").
3. **(Post-quantum) Zero-knowledge — intuition:** $\mathcal{V}$, and *a fortiori* any eavesdropping (Quantum) Probabilistic Polynomial-Time Algorithm ((Q)PPTA), cannot learn any information from the mere execution of the protocol.

We will now see in more detail the notion of (post-quantum) zero-knowledge of such an interactive proof system. As mentioned earlier, it is required to exhibit a simulator, that is, a (Q)PPTA, whose output should be indistinguishable from the transcript of a real execution of the protocol. One can find two very nice definitions by Lysyanskaya [Lys02] on both concepts of simulators and of black-box zero-knowledge. Let $\mathsf{View}_{\mathcal{A}}(\mathcal{A} \leftrightarrow \mathcal{B})$ be the view of $\mathcal{A}$ in its black-box interaction with $\mathcal{B}$.

Definition 1. (simulator) *A (Q)PPTA* S *is called a "simulator" for machine A's interactions with machine B on input x, if there exists a polynomial p such that for all auxiliary inputs a of length at most* $p(|x|)$*,*

$$\mathsf{View}_A(A(a,x) \leftrightarrow B_x) \approx \mathsf{View}_A(\mathsf{S}(x) \xrightarrow{\blacksquare} A(a,x)),$$

where $X \xrightarrow{\blacksquare} Y$ *denotes X black-box access to Y, and* $Y \approx Z$ *that Y and Z are statistically indistinguishable.*

Put differently, a (Q)PPTA S is said to be a "simulator" if the view from A of the black-box execution of a protocol is statistically indistinguishable from the output of S. Now, on the concept of black-box zero-knowledge:

Definition 2. (black-box zero-knowledge proof system) *A proof system* $(\mathcal{P}, \mathcal{V})$ *is a "black-box zero-knowledge proof system" for a language* $\mathbb{L}$ *if there exists a machine* S*, such that for all verifiers* $\mathcal{V}^*$ *and for all* $x \in \mathbb{L}$*,* S *is a simulator for* $\mathcal{P}(x)$*'s black-box interaction with* $\mathcal{V}^*$ *on input x.*

The classical way of building a simulator, given a $(\mathcal{P}, \mathcal{V})$ interactive proof system, is to decompose the verifier into black-box-access channels. For example, decomposing $\mathcal{V} = (\Phi_1, ..., \Phi_M)$, each channel Φ_i corresponds to the i-th interaction of the verifier on a given input. Then, using properties of black-box access, it is possible to design a simulator using techniques such as rewinding or out of order queries, for example.

3 Contributions

As mentioned earlier, our goal is to prove that any language that admits a zero-knowledge protocol with a constant-depth simulator must be in **BQP**. More formally, we will work at proving the following conjecture:

Conjecture 1. *Let $\mathbb{L}$ be any language admitting a black-box zero-knowledge interactive proof with a constant-depth simulator. Then $\mathbb{L} \in \boldsymbol{BQP}$.*

In this section are the first steps towards proving this goal.

3.1 Preliminary

A fundamental theorem in the field of quantum information is the No-Cloning Theorem. It was first formalized in 1982 by Wootters and Zurek [WZ82] (although it derives from Park's no-go result on measurements [Par70]), and it states that there exists no method $\mathcal{M}$, such that for an arbitrary quantum state $|\Psi\rangle$, $\mathcal{M}(|\Psi\rangle \otimes |0\rangle) = |\Psi\rangle \otimes |\Psi\rangle$. In other words, an arbitrary state cannot be cloned or copied with perfect precision. This property of quantum physics has been a defining rule in the field, and it is still used today as a cornerstone in mathematical proofs.

Thirty years later, Aaronson and Christiano [AC12], introduced the concept of subspace state, also known as *quantum money*. In essence, a subspace state $|S\rangle$ is the uniform superposition of all elements of an arbitrary subspace S of $\mathbb{Z}_2^\lambda$:

$$|S\rangle = \sum_{x \in S} \frac{1}{\sqrt{2^{|S|}}} |x\rangle \,. \tag{1}$$

Subspace states have two remarkable properties in particular:

1. Given their description (i.e., the hidden subspace used), they are easy to construct.
2. They are exponentially hard to clone (in λ) without the description, even with access to membership oracles.

Below is Aaronson and Christiano's result on the difficulty of cloning subspace states. Let $\mathcal{S}$ be the set of all $n/2$-dimensional subspaces of $\mathbb{F}_2^n$, and for $A \in \mathcal{S}$, let $V_A^{\otimes 2} = (|A\rangle \langle A|)^{\otimes 2}$ be the projector onto $|A\rangle^{\otimes 2}$.

Lemma 1. ([AC12]'s Theorem 25) *Let $A \leq \mathbb{F}_2^n$ be a uniformly-random element of $\mathcal{S}$. Then given one copy of $|A\rangle$, as well as oracle access to U_{A^*}, a counterfeiter C needs $\Omega(\sqrt{\varepsilon} 2^{n/4})$ queries to prepare a $2n$-qubit state ρ that $V_A^{\otimes 2}$ accepts with probability at least ε, for all $1/\varepsilon = o(2^{n/2})$. Here the probability is taken over the choice of $A \in \mathcal{S}$, as well as the behavior of C and $V_A^{\otimes 2}$.*

These two reasons make "quantum money" a very appropriate name for subspace states, as it designates a kind of *coin* or *bank note* which is hard to clone (assuming $|S| > log(\lambda)$), yet easily recognizable and constructible to whom knows its description. We will use subspace states extensively in the following.

The results we present now are our efforts at proving that there exists no post-quantum zero-knowledge interactive protocol with an efficient simulator. We use the term *efficient simulator* here to indicate a simulator with constant depth. The *depth* of a quantum circuit describes the number of steps from the beginning to the end, sequentially — the steps may individually run parallel operations, but the depth corresponds to the number of temporal "slices" one could make of the algorithm.

3.2 Towards proving Conjecture 1

Let us consider a zero-knowledge $(\mathcal{P}, \mathcal{V})$ interactive proof system with $\mathcal{V} = (U_1, ..., U_M)$. Each unitary gate U_i of $\mathcal{V}$ corresponds to the action of the verifier on its input at step i. Let us construct a second verifier $\mathcal{V}' = (U'_1, ..., U'_M)$, such that every U'_i is defined as follows:

$$U'_i = |S_{i+1}\rangle\langle S_i| \otimes U_i \otimes X + |S_i\rangle\langle S_{i+1}| \otimes U_i^\dagger \otimes X + \sum_{\substack{|x\rangle \perp |S_i\rangle \\ |x\rangle \perp |S_{i+1}\rangle}} |x\rangle\langle x| \otimes \mathbb{1} \otimes \mathbb{1}, \tag{2}$$

where $|S_1\rangle, ..., |S_M\rangle$ are random subspace states. One can easily verify that U'_i is unitary (and Hermitian), and that plugging in the input $\rho = |\Psi_i\rangle \otimes |\Phi_i\rangle \otimes |b_i\rangle$ (with b_i a binary value), performs the following operations linearly on each component of ρ:

- If $|\Psi_i\rangle = |S_i\rangle$, U'_i maps $|\Psi_i\rangle$ to $|S_{i+1}\rangle$, applies U_i to $|\Phi_i\rangle$ and turns $|b_i\rangle$ into $\left|\overline{b_i}\right\rangle$.
- If $|\Psi_i\rangle = |S_{i+1}\rangle$, U'_i maps $|\Psi_i\rangle$ to $|S_i\rangle$, applies $U_i^\dagger$ to $|\Phi_i\rangle$ and turns $|b_i\rangle$ into $\left|\overline{b_i}\right\rangle$.
- Otherwise, if $|\Psi_i\rangle$ is neither $|S_i\rangle$ nor $|S_{i+1}\rangle$, U'_i does nothing.

The input qubit $|b_i\rangle$ in $\rho = |\Psi_i\rangle \otimes |\Phi_i\rangle \otimes |b_i\rangle$ serves as an condition flag, signaling whether U_i and X are successfully applied or not. If U'_i turned $|b_i\rangle$ into $\left|\overline{b_i}\right\rangle$, then we consider it a *successful* query, otherwise it is a *failed* query.

By the definition of black-box zero-knowledge, there exists a simulator Sim, whose output is indistinguishable from the real execution of the protocol for all verifiers — let d be the depth of Sim. Notice that by construction, the interaction of $\mathcal{V}'$ with $\mathcal{P}$ is exactly the same as that of $\mathcal{V}$. Let $\mathsf{Transcript}(A \leftrightarrow B)$ be the function that returns the transcript of the interactions between A and B. Then:

Lemma 2. $\mathsf{Transcript}(\mathcal{V}' \leftrightarrow \mathcal{P}) \approx \mathsf{Transcript}(\mathcal{V} \leftrightarrow \mathcal{P})$.

Given that Sim is a zero-knowledge simulator, we also have the following:

Corollary 1. $\mathsf{Output}(\mathsf{Sim}(\mathcal{V}')) \approx \mathsf{Output}(\mathsf{Sim}(\mathcal{V}))$.

In other words, the transcript of the execution between $\mathcal{P}$ and $\mathcal{V}'$ is statistically indistinguishable from that of the execution involving $\mathcal{P}$ and $\mathcal{V}$.

Now let Sim_2 be a new simulator for $\mathcal{V}$, such that the additional operations of state swap and bit swap done by $\mathcal{V}'$, are offloaded from $\mathcal{V}'$ and performed by Sim_2 instead. The output of Sim_2 interacting with $\mathcal{V}$ and that of Sim interacting with $\mathcal{V}'$ are exactly the same, since the same additional actions are performed during the execution.

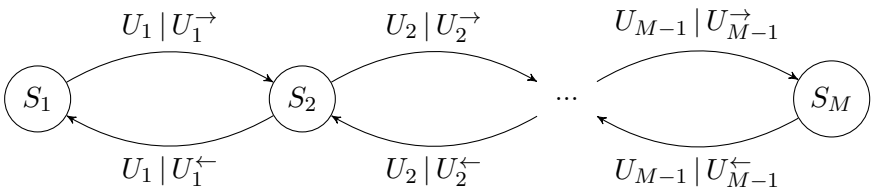


Figure 1: FSM (Finite State Machine)

Claim 3.1. $\mathsf{Output}(\mathsf{Sim}_2(\mathcal{V})) \approx \mathsf{Output}(\mathsf{Sim}(\mathcal{V}'))$.

Proof. The code executed by both algorithms is the same, therefore their output must be the same. □

Let FSM be the transducer depicted in Figure 1, that maps each input word to a series of $U_i^{\leftrightarrows}$ that we will define below. All states of the FSM are accepting states; the rejecting states and their transitions are omitted for clarity, but any transition not explicitly shown on the transducer leads to a trap state. We also define the two new types of gates $U_i^{\rightarrow}$ and $U_i^{\leftarrow}$ that the FSM uses:

- $U_i^{\rightarrow}$ ("U_i' forward") corresponds to applying U_i from state S_i, leading to state S_{i+1}.
- $U_i^{\leftarrow}$ ("U_i' backward") corresponds to applying U_i from state S_{i+1}, leading to S_i.

We can now define a third simulator Sim_3, which behaves as follows: for every interaction to the verifier, Sim_3 executes, at each step, the same operations as Sim_2 (that is, the U_i, and the bit and state swap). At the same time, it keeps track of the execution of the protocol on the FSM. When applying a gate U_i, Sim_3 can decide whether the path taken corresponds to $U_i^{\leftarrow}$ or to $U_i^{\rightarrow}$ by looking at its current position on the FSM. Depending on which $U_i^{\leftrightarrows}$ the path corresponds to, Sim_3 performs additional actions on the input:

- If the path corresponds to $U_i^{\rightarrow}$, before applying U_i, Sim_3 performs a projective measurement of the input on $\{|S_i\rangle\langle S_i|, \mathbb{1} - |S_i\rangle\langle S_i|\}$ (with respective outcomes 0 and 1), and only applies U_i if the outcome is 0. If the outcome is 1, Sim_3 aborts.
- If the path instead corresponds to $U_i^{\leftarrow}$, before applying U_i, Sim_3 performs a projective measurement of the input on $\{|S_{i+1}\rangle\langle S_{i+1}|, \mathbb{1} - |S_{i+1}\rangle\langle S_{i+1}|\}$ (notice the different projector) and only applies U_i if the outcome is 0 (or aborts if the outcome is 1).

These $U_i^{\leftrightarrows}$ gates are useful to make sure the U_i are applied with the desired inputs and "direction" (since the corresponding U_i' is also Hermitian). In addition to these new gates, if applying a U_i leads to the trap state on the FSM, Sim_3 aborts immediately.

Two properties are important to notice in the FSM we describe here. The first is that it is deterministic, meaning that there only exists a single path for any word, accepted or rejected. The second and most important property resides in the fact that for every

subspace state $|S_i\rangle$ provided to the simulator, it must be given back when applying the next gate in the sequence of execution. Each gate, forward or backwards, requires as input the subspace state given by the gate applied just before. If the state provided at time t does not correspond to the state that was returned at time $t-1$, Sim_3 aborts (before Sim_2).

To summarize:

- Sim is a simulator for $\mathcal{V}$ and $\mathcal{V}'$.
- Sim_2 is a simulator designed for $\mathcal{V}$, that takes the responsibility of doing the swap operations of $\mathcal{V}'$ on its side. Sim and Sim_2 are indistinguishable.
- Sim_3 is a simulator designed for $\mathcal{V}$ similar to Sim_2, with the additional tasks of applying a projective measurement and (possibly) aborting under specific conditions.

We assume here that Sim_2 and Sim_3 must be sequential simulators — that is, they can not successfully apply two U_{j_1}, U_{j_2} gates simultaneously. We will show later in Lemma 4 that this assumption is without loss of generality. We now argue that the outputs of Sim_2 and Sim_3 are also indistinguishable.

Lemma 3. $\mathsf{Sim}_3(\mathcal{V}) \approx \mathsf{Sim}_2(\mathcal{V})$

Proof. Since Sim_2 and Sim_3 apply the same gates in the same order, the only possible difference would be that Sim_3 aborts when Sim_2 does not. We will show that this can only happen with negligible probability. In other words, we will work at proving the following:

$$|\Pr[\mathsf{Sim}_3 \text{ aborts}] - \Pr[\mathsf{Sim}_2 \text{ aborts}]| \leq \mathsf{negl}(\lambda) \tag{3}$$

The proof is by contradiction. Let us suppose that $|\Pr[\mathsf{Sim}_3 \text{ aborts}] - \Pr[\mathsf{Sim}_2 \text{ aborts}]| \geq 1/p(\lambda)$ for some polynomial p. Two situations can lead Sim_3 to abort before Sim_2: first, Sim_2 can apply a gate that makes Sim_3 move on FSM to the trap state (case (a)), or second, Sim_3 can also abort if a projective measurement failed in a $U_i^{\rightleftarrows}$ (case (b)). We will show that in both situations, we can build an algorithm capable of cloning subspace-state-quantum money, hence contradicting Lemma 1.

First let us define some notions: we write $\mathbb{U} = \bigcup_i^M \{U_i^{\rightarrow}, U_i^{\leftarrow}\}$ to be the alphabet that contains all gates $U_i^{\rightleftarrows}$ available to Sim_3. Let $\pi \in \mathbb{U}^*$ be a word describing any sequence of gates, written in chronological order of execution. Let also $\tilde{\pi}$ be the longest prefix of π that Sim_3 can run successfully — finally, let $U_j^{\rightleftarrows}$ be the first gate in π, such that:

a) Sim_3 applies $\tilde{\pi}\,|\,U_j^{\rightleftarrows}$, making the FSM go to a trap state; or

b) A projective measurement fails with probability $1/q(\lambda)$, with q some polynomial, when Sim_3 applies $\tilde{\pi}\,|\,U_j^{\rightleftarrows}$,

both leading Sim_3 to abort. As a corollary, it is important to notice that:

a) No applied $U_k^{\rightleftarrows}$ during $\tilde{\pi}$ makes the FSM go to a trap state; and

b) The event where a projective measurement fails when applying any $U_k^{\rightleftarrows}$ during $\tilde{\pi}$ happens with negligible probability.

We can see that $U_j^{\rightleftarrows}$ corresponds to the "failure point" of Sim_3, and yet that the execution of Sim_2 corresponding to $\tilde{\pi} \,|\, U_j^{\rightleftarrows}$ is successful. Let us now define two adversaries $\mathbf{A}_1$ and $\mathbf{A}_2$ that are capable of cloning a quantum coin $|\$\rangle$ (one for each lose case). $\mathbf{A}_1$ and $\mathbf{A}_2$ receive the coin and oracle access to $\mathcal{O}_\$$ which lets them verify whether a state is $|\$\rangle$ or not.

'Trap state' situation (case (*a*)): Sim_3 will abort before Sim_2 if $\tilde{\pi} \,|\, U_j \notin \mathcal{L}(\mathsf{FSM})$. Such a situation can happen if U_j is applied from S_i, where $j > i$ or $j < i - 1$.

If Sim_2 can successfully run $\tilde{\pi} \,|\, U_j$, then the input state $|\Psi\rangle$ that Sim_2 provided must be either $|S_j\rangle$ or $|S_{j+1}\rangle$. $\mathbf{A}_1$ randomly chooses one of the two states and replaces it with the quantum coin. Without loss of generality, let us assume that the subspace state chosen is $|S_j\rangle$. The algorithm will execute Sim_2 on $\tilde{\pi}$ using the quantum coin $|\$\rangle$ in place of $|S_j\rangle$, which should complete successfully with probability $1/2 - \mathsf{negl}$ because the right state was picked uniformly at random among the two, and because the distributions of $|\$\rangle$ and $|S_j\rangle$ are identical. By construction of the FSM, and since $\tilde{\pi}$ is successfully executed by Sim_2 with probability half, $\mathbf{A}_1$ has the quantum coin in its possession with the same probability (say, in register $\mathcal{X}$). Just before executing U_j, the input provided by Sim_2 is intercepted by $\mathbf{A}_1$ and stored in register $\mathcal{Y}$. The algorithm immediately returns $\mathcal{X} \otimes \mathcal{Y}$, which projects onto $|\$\rangle \otimes |\$\rangle$ with at least polynomial probability, because $\mathbf{A}_1$ chose the right subspace state with probability $1/2$ (for $\mathcal{X}$), and because Sim_2 could successfully execute U_j with polynomial probability (for $\mathcal{Y}$).

Despite $U_j^{\rightleftarrows}$ being the first symbol of π that causes Sim_3 to abort, $\mathbf{A}_1$ or $\mathbf{A}_2$ may still have to simulate it with the quantum coin: for example, the corresponding gate $U_j = U_j^{\rightleftarrows}$ may have been applied before, forward and then backward, without causing a problem. Either algorithm can simulate the behavior of U_{j-1} and U_j using the quantum coin as described below:

- U_{j-1} is simulated with the following rules: $\mathbf{A}_1$ checks whether $|\Psi\rangle$ is $|\$\rangle$ using the oracle $\mathcal{O}_\$$. If $|\Psi\rangle = |\$\rangle$, $\mathbf{A}_1$ returns the subspace state $|S_{j-1}\rangle$. Otherwise, the algorithm checks whether $|\Psi\rangle = |S_{j-1}\rangle$ using a projective measurement, and if successful, it lends back the quantum coin $|\$\rangle$. Otherwise, if $|\Psi\rangle$ if neither $|S_{j-1}\rangle$ nor $|\$\rangle$, $\mathbf{A}_1$ aborts (as would Sim_2 do).

- U_j is simulated with similar rules: $\mathbf{A}_1$ checks whether the input state is $|S_{j+1}\rangle$. If $|\Psi\rangle = |S_{j+1}\rangle$, $\mathbf{A}_1$ lends the coin $|\$\rangle$. Otherwise, the algorithm checks whether the input state is $|\$\rangle$, in which case it returns the next subspace state $|S_{j+1}\rangle$. If $|\Psi\rangle$ is neither state, the algorithm aborts (again, as would Sim_2 do).

- When $\mathbf{A}_1$ needs to perform a projective measurement for $|\$\rangle$ (say, in the preliminary step of a $U_i^{\rightleftarrows}$), it queries the oracle $\mathcal{O}_\$$ to verify that the state is indeed $|\$\rangle$.

'Failed projection' situation (case (*b*)): Even if $\pi \in \mathcal{L}(\mathsf{FSM})$, Sim_3 can still abort before Sim_2 if the projective measurement associated to $U_i^{\rightarrow}$ or $U_i^{\leftarrow}$ fails. There are two

possibilities concerning its input state $|\Psi\rangle$ in case of such a failure:

1. If U_j corresponds to $U_j^{\rightarrow}$, $|\Psi\rangle \neq |S_j\rangle$, therefore $|\Psi\rangle$ must be $|S_{j+1}\rangle$.
2. If U_j corresponds to $U_j^{\leftarrow}$, $|\Psi\rangle \neq |S_{j+1}\rangle$, therefore $|\Psi\rangle$ must be $|S_j\rangle$.

We show that in either situation, we can build an algorithm $\mathbf{A}_2$ capable of breaking quantum money. $\mathbf{A}_2$ will behave differently depending on the gate at which the "failure point" is.

1. Case $U_j^{\rightarrow}$: before starting the game, $\mathbf{A}_2$ changes $|S_{j+1}\rangle$ with $|\$\rangle$ and executes the simulator on $\tilde{\pi}$. The gates U_j and U_{j+1}, as well as the projective measurement, are simulated using the same strategy as the one presented above.

   Again, the probability that Sim_3 executes successfully on $\tilde{\pi}$ is the same as that of Sim_2, because both distributions of $|\$\rangle$ and $|S_{j+1}\rangle$ are identical. Since $\tilde{\pi} \in \mathcal{L}(\mathsf{FSM})$, when Sim_2 reaches $U_j^{\rightarrow}$, $\mathbf{A}_2$ already has the quantum coin $|\$\rangle$ in its possession (say, in register $\mathcal{X}$), right at the end of executing $\tilde{\pi}$, by construction of the FSM. $\mathbf{A}_2$ intercepts the input provided by the simulator and stores it in register $\mathcal{Y}$. $\mathbf{A}_2$ then immediately returns with $\mathcal{X} \otimes \mathcal{Y}$, which projects onto $|\$\rangle \otimes |\$\rangle$ with probability $1/p(\lambda) - \mathsf{negl}$ because we assumed that the projective measurement failed with that probability.

2. Case $U_j^{\leftarrow}$: before starting the game, $\mathbf{A}_2$ changes $|S_j\rangle$ with $|\$\rangle$ using the same strategy as before and executes the simulator on $\tilde{\pi}$, which completes successfully with nonnegligible probability. Since U_j is considered as a backward gate, when the simulation has reached the last gate of $\tilde{\pi}$ (namely $U_{j-1}^{\rightarrow}$), the state that Sim_2 gave back must project on $|\$\rangle$ with at least polynomial probability (alternatively, the last gate could also be $U_{j+1}^{\leftarrow}$, in which case $\mathbf{A}_2$ had already received the coin). $\mathbf{A}_2$ stored it in register $\mathcal{X}$, returned $|S_{j+1}\rangle$ and proceeded to applying $U_j^{\leftarrow}$. On executing $U_j = U_j^{\leftarrow}$, $\mathbf{A}_2$ intercepts the input that Sim_2 provides and stores it in register $\mathcal{Y}$. The algorithm immediately returns with $\mathcal{X} \otimes \mathcal{Y}$, which again projects onto $|\$\rangle \otimes |\$\rangle$ with probability $1/p(\lambda) - \mathsf{negl}$.

Thus, if $\pi \notin \mathcal{L}(\mathsf{FSM})$, or if $\pi \in \mathcal{L}(\mathsf{FSM})$ but a projective measurement failed (i.e. the order of the $U_i^{\rightarrow}$ and $U_i^{\leftarrow}$ was not respected), we can clone quantum money. Since we know that this is not possible, we conclude that Lemma 3 is true, and therefore Sim_3 does not abort sooner than Sim_2 except with negligible probability. □

Proposition 1. $\mathsf{Output}(\mathsf{Sim}_3(\mathcal{V})) \approx \mathsf{View}_{\mathcal{V}}(\mathcal{V} \leftrightarrow \mathcal{P})$

Proof. By transitivity of the indistinguishability, and observing that the respective outputs of Sim and Sim_2 are statistically indistinguishable, as well as the outputs of Sim_2 and Sim_3, we conclude that Sim_3 is a valid simulator for the zero-knowledge interactive proof presented earlier for all verifiers. □

This proves that the order of execution, and the input provided by a sequential simulator, must be consistent with the language described by the transducer in Figure 1. We

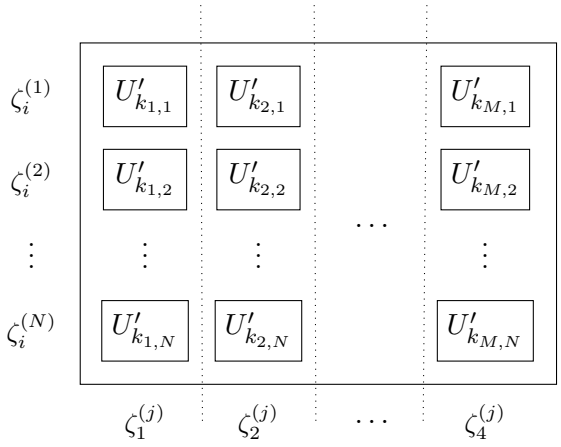


Figure 2: Diagram of a parallel simulator. Each gate shown here corresponds to a $\zeta_i^{(j)}$.

will now show that even in the context of a parallel simulator, it is not possible to successfully apply two gates U'_{i_1}, U'_{i_2} simultaneously, meaning that the number of *effective* queries to the verifier can be at most the depth of the circuit. We use the term "*effective* query" to describe a query that happens at run-time (for example in a successful controlled operation).

Lemma 4. Sim_3 *makes at most* d *effective queries to* $\mathcal{V}$*, that is,* Sim_3 *queries* $\mathcal{V}$ d *times or less during the execution.*

Proof. Again, the proof is by contradiction. We suppose that Sim_2 is a parallel simulator that can make two successful queries to $\mathcal{V}$ simultaneously, with probability $1/p(\lambda)$ (for some polynomial p). We denote ζ_i the step of the execution of Sim_2 at depth i, and $\zeta_i^{(j)}$ the j-th parallel execution of a U_k (for any k) on step ζ_i (as illustrated on Figure 2).

Let ζ_{i^*} $(1 \leq i^* \leq M)$ be the first depth of Sim_2 for which there exists j_1, j_2, k_1, k_2, such that $U_{k_1} = \zeta_{i^*}^{(j_1)}$ and $U_{k_2} = \zeta_{i^*}^{(j_2)}$ are both applied successfully with probability $1/p(\lambda)$. Consequently, this means that the execution of the simulator, up to ζ_{i^*-1} included, was (or could have been) sequential, since no depth $i' < i^*$ witnessed two gates applied successfully in parallel. In particular, the execution until ζ_{i^*-1} must respect the constraints presented in the proof of Lemma 3, and the corresponding word must be in the language of the FSM. Now assuming that two gates can be applied in parallel with nonnegligible probability, we will show that we can clone an arbitrary subspace state quantum coin.

- **Case** $k_1 < k_2 - 1$ **or** $k_2 < k_1 - 1$**.** We proved in Lemma 3 that this is not possible without applying other gates in between.

- **Case** $k_1 = k_2 - 1$ **or** $k_2 = k_1 - 1$**.** Without loss of generality, let us assume that $k_1 = k_2 - 1$. Since we cannot know in advance in which "direction" U_{k_1} and U_{k_2} were applied, we define four sub-algorithms $\mathbf{A}_{3,1}, ..., \mathbf{A}_{3,4}$, which will be picked at random with uniform probability. As the first steps, $\mathbf{A}_{3,1}, ..., \mathbf{A}_{3,4}$ all replace $|S_{k_2}\rangle$

with $|\$\rangle$ using the strategy presented before, and execute the simulator until ζ_{i^*-1}. Their strategy differs now:

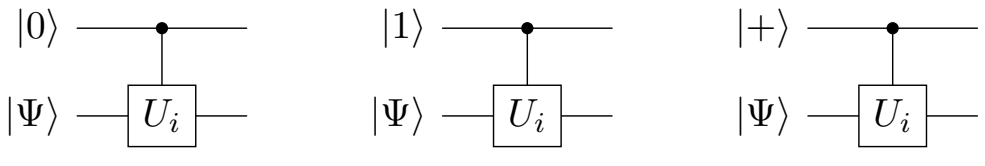
1. **Subcase $U_{k_1}^{\rightarrow}$, $U_{k_2}^{\rightarrow}$:** Algorithm $\mathbf{A}_{3,1}$ applies U_{k_1}, intercepts the output and stores it in register $\mathcal{X}$. At the same time, it intercepts the input of U_{k_2} and stores it in $\mathcal{Y}$. Finally, $\mathbf{A}_{3,1}$ returns $\mathcal{X} \otimes \mathcal{Y}$.
2. **Subcase $U_{k_1}^{\rightarrow}$, $U_{k_2}^{\leftarrow}$:** Algorithm $\mathbf{A}_{3,2}$ applies U_{k_1} and U_{k_2}, intercepts their outputs and stores them respectively in registers $\mathcal{X}$ and $\mathcal{Y}$. $\mathbf{A}_{3,2}$ returns $\mathcal{X} \otimes \mathcal{Y}$.
3. **Subcase $U_{k_1}^{\leftarrow}$, $U_{k_2}^{\rightarrow}$:** Before applying U_{k_1} and U_{k_2}, algorithm $\mathbf{A}_{3,3}$ intercepts their inputs and stores them respectively in registers $\mathcal{X}$ and $\mathcal{Y}$. $\mathbf{A}_{3,3}$ returns $\mathcal{X} \otimes \mathcal{Y}$.
4. **Subcase $U_{k_1}^{\leftarrow}$, $U_{k_2}^{\leftarrow}$:** Algorithm $\mathbf{A}_{3,4}$ applies U_{k_2}, intercepts the output and stores it in register $\mathcal{X}$. At the same time, it intercepts the input of U_{k_1} and stores it in $\mathcal{Y}$. $\mathbf{A}_{3,4}$ returns $\mathcal{X} \otimes \mathcal{Y}$.

If the right algorithm is picked, $\mathcal{X} \otimes \mathcal{Y}$ projects onto $|\$\rangle \otimes |\$\rangle$ with probability $1/p(\lambda)$ because we supposed that Sim_2 can run U_{k_1} and U_{k_2} with this probability. The right choice is made with probability 1/4, therefore the probability of cloning quantum money is still polynomial.

- **Case $k_1 = k_2$.** There are four possibilities in this case, depending on the input provided to each gate. We define four new sub-algorithms $\mathbf{A}_{4,1}, ..., \mathbf{A}_{4,4}$ for each sub case, picked at random during the execution:
  1. **Subcase $|S_{k_1}\rangle \otimes |S_{k_1}\rangle$:** $\mathbf{A}_{4,1}$ replaces $|S_{k_1}\rangle$ with $|\$\rangle$, and executes the simulator normally until ζ_{i^*-1}. On reaching ζ_{i^*}, $\mathbf{A}_{4,1}$ intercepts the two inputs provided by Sim_2 and stores them in $\mathcal{X}$ and $\mathcal{Y}$. $\mathbf{A}_{4,1}$ returns $\mathcal{X} \otimes \mathcal{Y}$.
  2. **Subcase $|S_{k_1+1}\rangle \otimes |S_{k_1+1}\rangle$:** $\mathbf{A}_{4,2}$ behaves essentially the same as $\mathbf{A}_{4,1}$, but replaces $|S_{k_1+1}\rangle$ with $|\$\rangle$.
  3. **Subcase $|S_{k_1+1}\rangle \otimes |S_{k_1}\rangle$:** $\mathbf{A}_{4,3}$ replaces $|S_{k_1+1}\rangle$ with $|\$\rangle$ and executes the simulator until ζ_{i^*-1}. Before applying U_{k_2}, $\mathbf{A}_{4,3}$ intercepts the corresponding input and stores it in register $\mathcal{X}$. $\mathbf{A}_{4,3}$ then applies U_{k_1}, intercepts the output and stores it in $\mathcal{Y}$. It then return $\mathcal{X} \otimes \mathcal{Y}$.
  4. **Subcase $|S_{k_1}\rangle \otimes |S_{k_1+1}\rangle$:** $\mathbf{A}_{4,4}$ behaves essentially the same as $\mathbf{A}_{4,3}$, but intercepts the output of U_{k_2} and the input of U_{k_1} instead of the other way around.

  The probability that the right algorithm is chosen is also 1/4. Since we supposed that U_{k_1} and U_{k_2} are successfully run with probability $1/p(\lambda)$, $\mathcal{X} \otimes \mathcal{Y}$ still projects onto $|\$\rangle \otimes |\$\rangle$ with polynomial probability.

During the execution of the simulator until ζ_{i^*-1}, it may be necessary to simulate the unitaries involved with the quantum coin injected by $\mathbf{A}_3$ or $\mathbf{A}_4$. This is also achieved using a strategy similar to the one presented before. We stress that by our initial supposition,



(a) (b) (c)

Figure 3: Some variations of the control qubit in a controlled-U_i circuit

no algorithm will have to simulate two gates involved with the coin simultaneously, since ζ_{i^*} must be the first gate with two gates applied successfully in parallel. In addition, again, the probability that the execution until ζ_{i^*} happens without failure is still the same, because the distribution of the quantum coin and the subspace state it replaced are identical.

Hence, since Sim_3 has depth d and cannot make two successful queries to $\mathcal{V}$ in parallel, then the number of effective queries to $\mathcal{V}$ is at most d. Thus, for any zero-knowledge protocol where the corresponding simulator has constant depth d, we have defined a simulator which makes at most d effective queries to the verifier $\mathcal{V}$. □

4 Discussion and future work

Our work on the conjecture ends here. There is still a few steps to prove Conjecture 1. Unfortunately, we lacked enough time to explore these steps. In this section, we will discuss some important questions for the future of this work. Namely, we want to discuss the question of counting the queries to $\mathcal{V}$ and the next steps in our attempt at proving the conjecture.

4.1 Counting the queries to $\mathcal{V}$

Our current work proves that for any zero-knowledge interactive protocol that admits a constant-depth simulator, we can always construct a simulator with a constant amount of *effective* queries to $\mathcal{V}$.

We make the distinction between counting "queries" and counting "*effective* queries", because the current work does not describe how to build a simulator where the number of queries to $\mathcal{V}$ is at most d — instead, it shows how to build a simulator where the number of queries to $\mathcal{V}$, *during the execution*, is at most d.

Although this precision may seem superfluous in the classical context (whether a query is performed or not is binary), it is less easy to tell when taking a quantum perspective, where qubits can be a superposition of several states. A most obvious one is $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$, where $|0\rangle$ and $|1\rangle$ are in uniform superposition in the state.

Let us go through an example to illustrate why this difference is important. Figure 3 shows three variations of a quantum circuit with a controlled-U_i, where the control qubit respectively takes the values $|0\rangle$, $|1\rangle$ and $|+\rangle$. More formally, this circuit can be generally

expressed as

$$W = |1\rangle\langle 1| \otimes U_i + |0\rangle\langle 0| \otimes \mathbb{1},$$

with input state $|c\rangle \otimes |\Phi\rangle$, where $|c\rangle$ is the control qubit and $|\Phi\rangle$ is the target state. Intuitively, one could say that the situation shown in Figure 3a does not constitute a query to U_i, because the control qubit is set on $|0\rangle$, meaning that the U_i will not be applied. On the contrary, Figure 3b and Figure 3c should count as one query each, because they both have $|1\rangle$ as the control qubit, or at least as a superposed component. Looking at the mathematical expression of W and the figure, it can indeed appear natural to only account for *effective* queries when counting queries.

However, quantum query complexity generally considers that a controlled-U_i operation always counts as a query to U_i, even if it is trivial and results in no action on the target state. Hence, it does not matter whether applying the controlled gate actually results in applying U_i, because a controlled operation still counts as a (wasted) query.

Our work shows that Sim_3 has a constant number of *effective* queries to $\mathcal{V}$, but not of trivial queries. When introducing $\mathcal{V}'$, we indeed defined its unitary gates as controlled operations, acting on the target states depending on a control subspace state. In the rest of the proof, we focused on *successful* queries to these gates and chose not to count failed queries. As such, Lemma 4 concludes that Sim_3 makes a constant number of *effective* queries to the verifier.

4.2 Continuation of our work

4.2.1 From "effective queries" to "queries"

As far as possible, the first step towards improving our work would be to prove that we can build a simulator with at most d queries to the verifier $\mathcal{V}$ (that is, including trivial queries).

A possible strategy would be to build a new verifier for Sim_3, that creates a bridge between effective queries and queries. For example, let $\mathcal{V}^*$ be a third verifier for the zero-knowledge protocol mentioned earlier. By the definition of black-box zero-knowledge in Definition 2, Sim is a valid simulator for $\mathcal{V}^*$, and by extension, Sim_3 is also a simulator for $\mathcal{V}^*$. $\mathcal{V}^*$ takes as input $(k, |\sigma\rangle)$, where k corresponds to the index of the operation that the simulator wishes to execute on its input σ. Hence, calling $\mathcal{V}^*(k, |\sigma\rangle)$ corresponds to applying $U_k |\sigma\rangle$. With that in hands, we could prove that the number of queries is constant.

4.2.2 Proving that $\mathbb{L} \in \mathsf{BQP}$

Another step towards improving our work consists of the second part of the conjecture: it is prove that given a ZK interactive proof for a language $\mathbb{L}$, if the protocol admits a simulator with a constant number of queries to the verifier then $\mathbb{L} \in \mathrm{BQP}$.

The intuition behind the proof is the following: the simulator is a polynomial time algorithm, therefore is not capable of solving NP-hard problems given limited amounts of computational power. Where a classical simulator would use rewinding to get a satisfying

response from $\mathcal{V}$,[2] and would be able to output an indistinguishable transcript (we recall that the simulator does not have access to a valid witness, and needs to "cheat" somehow), this strategy is not available to a quantum simulator due to the No-Cloning Theorem.

A possible solution would be to use a technique similar to the one presented by Chia et al. [Chi+21]. The idea is that if we assume that there exists a simulator S with a constant number of queries to the verifier, then S does not need to obtain a specific, accommodating response from $\mathcal{V}$ in order to produce an indistinguishable transcript. This implies that S is itself an algorithm for the decision problem of membership to $\mathbb{L}$, therefore $\mathbb{L} \in \mathrm{BQP}$ since we assumed that S is a polynomial-time algorithm.

5 Conclusion

This report presents our contributions to the open question presented earlier, namely the characterization of languages that admit a constant-depth simulator. We show that for any post-quantum zero-knowledge protocol that admits a constant-depth simulator, we can build a second simulator which makes a constant number of effective queries to the verifier. In addition, we propose a strategy and the intuition for the next part of the proof, that is:

1. Proving that from a simulator with a constant number of *effective* queries, we can build a verifier that enables constant numbers of queries.

2. From that, showing that any language $\mathbb{L}$ that admits a post-quantum zero-knowledge protocol with a constant number of queries to the verifier, necessarily belongs in **BQP**.

3. As a result, any language that admits a constant-depth simulator must be in **BQP**.

This internship was extremely interesting and formative to me. In particular, I want to express my sincere gratitude to my supervisors for their time, guidance and availability. I especially enjoyed our weekly brainstorming sessions, where they showed patience and kindness, and where the most progress was made.

I also want to thank the ALMASTY and QI teams, who always made me feel welcome and included, and with whom I shared extraordinary moments during those six months. I feel privileged to continue in academics at the LIP6 and to be able to work among people like them.

[2] This is possible because we assume black-box access to $\mathcal{V}$. For example, in the classical setting, the simulator goes back before the commitment step if the challenge received is not accommodating (in the sense that it is not compatible with the false witness committed, meaning that the rest of the simulation will not be indistinguishable).

References

[AC12] Scott Aaronson and Paul Christiano. "Quantum money from hidden subspaces". In: *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*. STOC '12. New York, New York, USA: Association for Computing Machinery, 2012, pp. 41–60. ISBN: 9781450312455. DOI: 10.1145/2213977.2213983. URL: https://doi.org/10.1145/2213977.2213983.

[AG22] Prabhanjan Ananth and Alex B. Grilo. *Post-Quantum Zero-Knowledge with Space-Bounded Simulation*. 2022. arXiv: 2210.06093 [quant-ph]. URL: https://arxiv.org/abs/2210.06093.

[Chi+21] Nai-Hui Chia et al. *On the Impossibility of Post-Quantum Black-Box Zero-Knowledge in Constant Rounds*. 2021. arXiv: 2103.11244 [cs.CR]. URL: https://arxiv.org/abs/2103.11244.

[GK96] Oded Goldreich and Ariel Kahan. "How to construct constant-round zero-knowledge proof systems for NP". en. In: *Journal of Cryptology* 9.3 (June 1996), pp. 167–189. ISSN: 1432-1378. DOI: 10.1007/BF00208001.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The knowledge complexity of interactive proof-systems". In: *Proceedings of the seventeenth annual ACM symposium on Theory of computing*. STOC '85. New York, NY, USA: Association for Computing Machinery, Dec. 1985, pp. 291–304. ISBN: 978-0-89791-151-1. DOI: 10.1145/22145.22178. URL: https://dl.acm.org/doi/10.1145/22145.22178.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The Knowledge Complexity of Interactive Proof Systems". In: *SIAM Journal on Computing* 18.1 (Feb. 1989), pp. 186–208. ISSN: 0097-5397. DOI: 10.1137/0218012.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. "Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems". In: *J. ACM* 38.3 (July 1991), pp. 690–728. ISSN: 0004-5411. DOI: 10.1145/116825.116852. URL: https://doi.org/10.1145/116825.116852.

[Lys02] Anna Lysyanskaya. "Signature schemes and applications to cryptographic protocol design". eng. Accepted: 2005-10-14T19:34:05Z. Thesis. Massachusetts Institute of Technology, 2002. URL: https://dspace.mit.edu/handle/1721.1/29271.

[Par70] James L. Park. "The concept of transition in quantum mechanics". In: *Foundations of Physics* 1 (1970), pp. 23–33. URL: https://api.semanticscholar.org/CorpusID:55890485.

[WZ82] W. K. Wootters and W. H. Zurek. "A single quantum cannot be cloned". en. In: *Nature* 299.5886 (Oct. 1982), pp. 802–803. ISSN: 1476-4687. DOI: 10.1038/299802a0.