Uncloneable Quantum Advice

Anne Broadbent
University of Ottawa
abroadbe@uottawa.ca

Martti Karvonen
University of Ottawa
martti.karvonen@uottawa.ca

Sébastien Lord
University of Ottawa
sebastien.lord@uottawa.ca

Abstract

The famous no-cloning principle has been shown recently to enable a number of uncloneable functionalities. Here we address for the first time *unkeyed* quantum uncloneablity, via the study of a complexity-theoretic tool that enables a computation, but that is natively unkeyed: *quantum advice*. Remarkably, this is an application of the no-cloning principle in a context where the quantum states of interest are *not* chosen by a random process. We show the unconditional existence of promise problems admitting *uncloneable quantum advice*, and the existence of languages with uncloneable advice, assuming the feasibility of quantum copy-protecting certain functions. Along the way, we note that state complexity classes, introduced by Rosenthal and Yuen (ITCS 2022) — which concern the computational difficulty of synthesizing sequences of quantum states — can be naturally generalized to obtain state *cloning* complexity classes. We make initial observations on these classes, notably obtaining a result analogous to the existence of undecidable problems.

Our proof technique establishes the existence of *ingenerable sequences of finite bit strings* — essentially meaning that they cannot be generated by any uniform circuit family. We then prove a generic result showing that the difficulty of accomplishing a computational task on uniformly random inputs implies its difficulty on any fixed, ingenerable sequence. We use this result to derandomize quantum cryptographic games that relate to cloning, and then incorporate a result of Kundu and Tan (arXiv 2022) to obtain uncloneable advice. Applying this two-step process to a monogamy-of-entanglement game yields a promise problem with uncloneable advice, and applying it to the quantum copy-protection of pseudorandom functions with super-logarithmic output lengths yields a language with uncloneable advice.

Contents

1 Introduction

The no-cloning principle has been a key tenet of quantum information theory from its very early days [Die82, WZ82].[1] This principle tells us that it is, in general, impossible to create a perfect copy of an *unknown* quantum state. In these works, the state is unknown in the sense that it is selected uniformly at random from two possibilities. Follow up results, *e.g.*: [BH96, BDE+98, Wer98], also give bounds on the ability to create close-but-imperfect copies of states chosen from some set.

Quantum cryptography, which seeks to leverage quantum mechanical phenomena to achieve cryptographic goals, has greatly benefited from the no-cloning principle. Indeed, this principle establishes a stark qualitative difference between classical information, which can be perfectly copied, and quantum information, which cannot. In its most generic cryptographic application, it implies that a malicious eavesdropper cannot, in general, keep a perfect transcript of the quantum communication between two honest parties. This idea can be understood as being at the core of the security of many quantum cryptographic schemes, such as prepare-and-measure quantum key distribution protocols, *e.g.*: [BB84], and quantum money schemes, *e.g.*: [Wie83].

This leads to a fruitful avenue of research in quantum cryptography, occasionally known as *uncloneable cryptography*, which can be summarized by the following question: *Which notions from classical information processing can be made "uncloneable" by the addition of quantum mechanics and its no-cloning principle?* One such example, as initially set-out by Aaronson [Aar09], is the task of copy-protecting a function or a program. Quantum copy-protection, for a family of functions $\mathcal{F}$, can be understood as the process of encoding a member f of $\mathcal{F}$ as a quantum state ρ_f such that the following are satisfied:

- Correctness: There exists an honest procedure which, on input of ρ_f and x, outputs $f(x)$.
- Security: It is infeasible to "split" ρ_f into two quantum systems, both allowing the evaluation of f using any, possibly malicious, procedures.

Clearly, this is impossible to achieve in a purely classical setting but the no-cloning principle opens the door to achieving this in the quantum setting. Since Aaronson's introduction of this idea, there has been a flurry of works considering the question of quantum copy-protection, *e.g.*: [ALL+21, CMP20], and the closely related notion of secure software leasing, *e.g.*: [ALP21, BJL+21].

A key aspect of these existing constructions is that their security guarantees only hold if the function which is encoded is chosen at random from some larger set and not disclosed to the end-user. More formally, the family $\mathcal{F}$ is often understood as being the set of all maps $f(k, \cdot)$ generated from a single keyed function $f : \{0,1\}^\kappa \times \{0,1\}^d \to \{0,1\}^c$. The recipient of the copy-protected program then receives the state $\rho_{f(k,\cdot)}$ for a random key k which is unknown to them. More to the point, current approaches to quantum copy-protection cannot be applied to specific, unkeyed, functionalities. For example, it is unclear if, using existing techniques, it would be possible to copy-protect a *specific* algorithm A solving some fixed and known decision problem P.

In this work, we initiate the study of copy-protecting unkeyed functionalities, *i.e.*: copy-protecting fixed functions which are not chosen at random from a larger set. We frame our main results as the construction of *uncloneable quantum advice* for certain promise problems and languages. This framing occurs naturally since advice can already be understood as a program helping a user to solve a decision problem [Wat09].

[1]Diek, Wooters, and Zurek proved the no-cloning principle to argue that a proposed protocol for faster-than-light communication [Her82] was unphysical. However, a proof by Parks [Par70] of this principle already existed in the literature, albeit with no explicit connections to quantum information theory. See the work of Ortigoso [Ort18] for more on the history of the no-cloning principle.

One drawback of our work is that the resulting advice is either uncomputable, or *extremely* difficult to generate. We leave addressing this issues to future work.

Organization of this work. We complete our introduction by giving a high-level overview of our contributions in Section 1.1 and highlighting some open questions in Section 1.2. We then proceed to review some more technical preliminaries in Section 2 and formalize our novel technical tool, ingenerable sequences, in Section 3. We then examine cloning complexity classes in Section 4 before continuing with the main goal of this work in Section 5. In this last section, we briefly review quantum copy-protection, formally define uncloneable advice, and show how to instantiate this notion for one class of promise problems and one class of languages.

1.1 Contributions

We review here the main contributions in this work. We note that, for pedagogical reasons, this high-level review does not follow the same order as our detailed presentation.

1.1.1 Uncloneable Quantum Advice

Our first conceptual contribution is the formal definition of *uncloneable quantum advice*.[2] Defining this notion is the content of Section 5.2.

The prototypical complexity class with *quantum* advice is denoted **BQP**/qpoly. This class contains precisely the decision problems $P = (P_{\text{yes}}, P_{\text{no}})$ for which there exists a fixed sequence of quantum states $(\rho_n)_{n\in\mathbb{N}}$ and an efficient (polynomial-time) family of quantum circuits $(C_n)_{n\in\mathbb{N}}$ such that on input of a problem instance $p \in P$ and the corresponding advice state $\rho_{|p|}$, the circuit $C_{|p|}$ accepts (respectively, rejects) with probability at least $2/3$ if $p \in P_{\text{yes}}$ (respectively, $p \in P_{\text{no}}$). The "poly" in "qpoly" simply signifies that the advice states can contain at most a polynomial number of qubits. The "q", of course, stands for "quantum". Quantum advice was first introduced in [NY04] and this notion was further developed in [Aar05].

An important aspect of advice is that it is a fundamentally *non-uniform* resource and that there is no restriction on the computational complexity of generating it. In fact, the advice states may even encode the solutions to uncomputable problems.

In a survey of quantum complexity theory [Wat09], Watrous states that "[q]uantum advice is a formal abstraction that addresses this question: How powerful is quantum software?" Indeed, the circuit family $(C_n)_{n\in\mathbb{N}}$ from the previous paragraph can be understood as simply outputting the single bit which results from "running the software" encoded in the advice state $\rho_{|p|}$ with the problem instance p as the input. More formally, the decision problem P naturally defines a map $P_{\text{yes}} \cup P_{\text{no}} \to \{0, 1\}$ where a problem instance p is mapped to 1 if it is an element of P_{yes} and mapped to 0 if it is an element of P_{no}. Thus, ρ_n can be seen as a quantum program which allows a holder to correctly evaluate this map on all inputs of length n.

Since we have established a close parallel between quantum advice and quantum programs, it is natural that our definition of *uncloneable* quantum advice should follow closely the definition of

[2]There has been an ongoing implicit debate in the literature on the proper spelling of the word *uncloneable*. Many authors, perhaps even a majority, prefer *unclonable*, *i.e.* they omit the *e* from *clone* before appending the *-able* suffix. While the *Oxford English Dictionary* recognizes both *clonable* and *cloneable* (notably, neither appears in this reference with the *un-* prefix), it expresses a preference for *clonable* [OED22]. However, to the best of our knowledge, the earliest work in the field of quantum cryptography which uses the word *uncloneable* as an element of technical terminology [Got03] keeps the *e*. We will continue to follow this convention.

Note that this debate is not limited to the field of quantum cryptography. See, for example, Maes' discussion on this subject in their textbook on physically uncloneable functions [Mae13, sec. 2.3.1].

uncloneable quantum programs, as formalized by quantum copy-protection.

More precisely, we will say that the quantum advice states $(\rho_n)_{n\in\mathbb{N}}$ for some decision problem P are *uncloneable* if no efficient adversary A can split the advice state ρ_n between two other adversaries B and C such that both of them can simultaneously and correctly solve independently sampled problem instances p_B and p_C of length n with more than a negligible advantage over $\frac{1}{2}$. This is precisely the security requirement that a copy-protection scheme would aim to provide for the map $P_{\text{yes}} \cup P_{\text{no}} \to \{0,1\}$ described above. We specify here that the problem instances p_B and p_C are sampled uniformly at random among all *yes* instances with probability $\frac{1}{2}$ and uniformly at random among all *no* instances with probability $\frac{1}{2}$. Formalizing this, as we have done in Definition 29 of Section 5.2, yields a complexity class which we denote **neglQP**/upoly, where the "u" denotes the fact that the advice is uncloneable. The class **neglQP**, for its part, is precisely the class of problems which can be solved with negligible error in quantum polynomial time. As we now sketch, we define **neglQP**/upoly and not simply **BQP**/upoly because it is possible that these two classes are distinct.

By the standard amplification technique of repeating a computation a polynomial number of times and taking a majority vote on the result, we see that **neglQP** = **BQP**. In fact, this same argument also yields **neglQP**/qpoly = **BQP**/qpoly, provided that polynomially many copies of the basic advice state are provided. However, it is unclear if **neglQP**/upoly = **BQP**/upoly, since the basic amplification technique previously sketched fails to yield this equality. Indeed, if polynomially many copies of an advice state are provided, half can be given to one party and half to another. This breaks the uncloneability guarantee. It is also unclear if the other common amplification technique, given by Marriot and Watrous [MW05], yields this equality between complexity classes. Indeed, their technique does not require sending many copies of the advice state, but it does require changing the advice and there is no *a priori* guarantee that the uncloneability guarantee is preserved under this transformation.

1.1.2 Cloning Complexity Classes

As a second conceptual contribution, we define *cloning complexity classes.* The uncloneability guarantee for advice states discussed in the previous section is *not* simply that it is impossible to efficiently implement the transformation $\rho_n \mapsto \rho_n \otimes \rho_n$. Indeed, if it was possible to efficiently transform the advice state ρ_n into $\sigma_n \otimes \sigma_n$, where $\sigma_n \neq \rho_n$ but where each σ_n would still allow malicious users to correctly solve the decision problem, then ρ_n fails to be uncloneable advice in the sense described in Section 1.1.1. In fact, this distinction between the (in)ability to copy a quantum state and the (in)ability of copying its underlying operational capabilities is at the source of many interesting aspects and challenges of uncloneable cryptography.

Nonetheless, it is evident that the infeasibility of implementing the transformation $\rho_n \mapsto \rho_n \otimes \rho_n$ is a necessary condition for the sequence $(\rho_n)_{n\in\mathbb{N}}$ to be uncloneable advice for some decision problem. This naturally leads us to consider questions of the following form, with a particular interest in cases where the answer is negative:

> *Given a sequence $(\rho_n)_{n\in\mathbb{N}}$ of fixed quantum states, is there a sequence of circuits $(C_n)_{n\in\mathbb{N}}$, satisfying some given computational constraints, such that $C_n(\rho_n) \approx \rho_n \otimes \rho_n$ for all n?*

Our insight here, covered in Section 4, is that this type of question generalizes those captured by the *state complexity classes* recently studied by Metger, Rosenthal, and Yuen [RY21, MY23]. To take an explicit example of such a class presented in these works, a sequence of states $(\rho_n)_{n\in\mathbb{N}}$ is in the state complexity class **statePSPACE** if and only if there exists a uniform polynomial-space

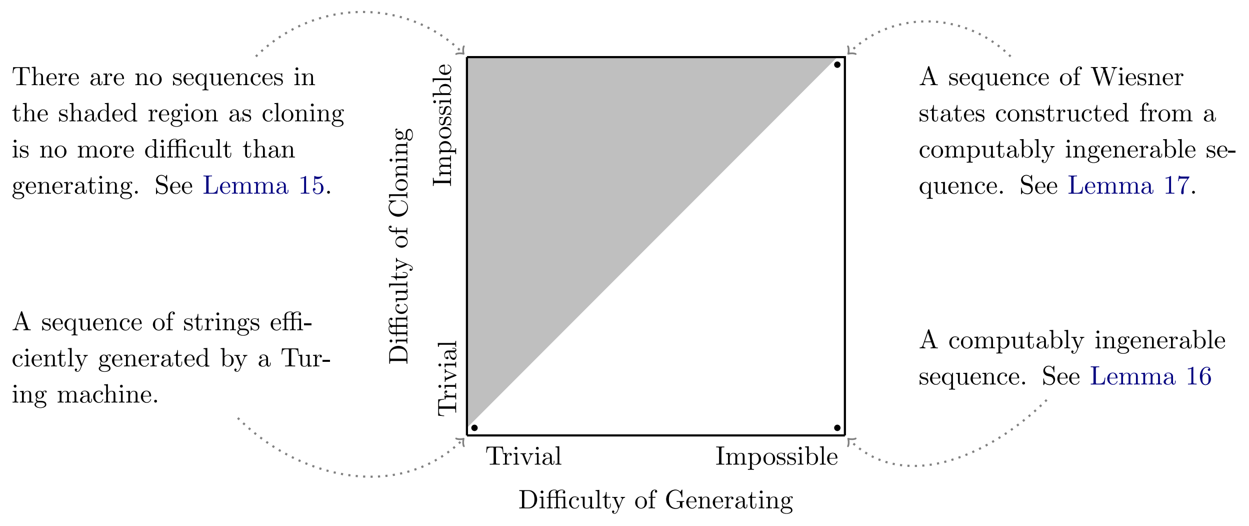

There are no sequences in the shaded region as cloning is no more difficult than generating. See Lemma 15.
A sequence of strings efficiently generated by a Turing machine.
A sequence of Wiesner states constructed from a computably ingenerable sequence. See Lemma 17.
A computably ingenerable sequence. See Lemma 16
Difficulty of Cloning
Impossible
Trivial
Trivial
Impossible
Difficulty of Generating

Figure 1: An annotated and informal representation of the space of sequences of quantum states contrasting the difficulty to generate and to clone a given sequence. Ingenerable sequences, which are novel to this work, are discussed in more details in Section 1.1.3.

circuit family $(C_n)_{n\in\mathbb{N}}$ such that C_n outputs a state overwhelmingly close to ρ_n on the empty input. The questions of the form above can be recast to form *cloning complexity classes.* These classes can then be understood as the "$1 \to 2$" generalization of the "$0 \to 1$" state complexity classes. We formalize this in Definition 14.

We note in passing that existing works on the no-cloning principle do not answer cloning complexity questions of the form presented above as they all consider the difficulty of cloning a state sampled at random from some larger set. Our cloning complexity classes explicitly remove this randomness from the cloning task.

The crux of this work is not focused on cloning complexity classes, but the tools and techniques we developed to instantiate uncloneable quantum advice do yield two interesting results on these classes and how they relate to state complexity classes:

- There exists a sequence of fixed quantum states which cannot be *generated* by any uniform circuit family, but which can be cloned by an efficient circuit family (Lemma 16).
- There exists a sequence of fixed quantum states which cannot be *cloned* by any uniform circuit family, even those implementing arbitrarily large computations (Lemma 17).

These two results, with an additional generic result showing that cloning cannot be more difficult than generating (Lemma 15) are visualized in Figure 1 which locates sequences of states with respect to two axes: one denoting the difficulty to generate them and one denoting the difficulty to clone them. Moreover, Lemma 17 can be interpreted as establishing a distinction between cloneable and uncloneable sequences of states, similar to how maps can be partitioned between those which are computable and those which are not. We leave the study of complexity-theoretic analogues (*e.g.*: establishing distinctions between efficiently cloneable and uncloneable sequences) for future work.

We emphasize that the development and study of cloning complexity classes are not the main goal of this work; however, they do offer an interesting way for us to connect some aspects of our work with existing results. We also believe that the study of cloning complexity classes could be of central interest and importance for further developments in uncloneable cryptography.

1.1.3 Ingenerable Sequences of Bit Strings

The main technical tool which we develop in this work are *ingenerable sequences* of bit strings. This is the content of Section 3.

In a nutshell, a sequence $(s_n)_{n\in\mathbb{N}}$ of bit strings $s_n \in \{0,1\}^*$ is computably (respectively, exponential-time) ingenerable if every uniform (respectively, exponential-time) circuit family $(C_n)_{n\in\mathbb{N}}$ eventually always fails at generating the elements of the sequence with sufficiently large probability. More precisely, there exists a polynomial p such that for every uniform (respectively, exponential-time) family of quantum circuits $(C_n)_{n\in\mathbb{N}}$ the quantity $\langle s_n| C_n(\varepsilon) |s_n\rangle$, where ε denotes the empty input, is eventually strictly smaller than $p(n) \cdot 2^{-|s_n|}$. We emphasize that the polynomial p is universal, in the sense that it does not depend on the circuit family $(C_n)_{n\in\mathbb{N}}$. However, the value of n at which point the inequality begins to be satisfied can change from one circuit family to the next.

A simple counting argument demonstrates that such sequences exist unconditionally for many desired lengths of the component strings. While computably ingenerable sequence are by their nature uncomputable, we also show that certain exponential-time ingenerable sequence can be computed by classical deterministic Turing machines in triple exponential time. Our counting argument is constructive, in the sense that it uniquely describes an ingenerable sequence, even in the case of computably ingenerable sequences where we cannot actually compute their elements. Further, we give an explicit classical deterministic algorithm running in triple exponential time which computes an exponential-time ingenerable sequence. These results are stated explicitly in Theorem 9.

Note that ingenerable sequences are purely classical information and so they can trivially be copied, this immediately yields Lemma 16 which was discussed previously in Section 1.1.2.

Ingenerable sequences are useful for us in this work due to a generic result, Theorem 10, relating the difficulty of a computational task on uniformly random inputs to its difficulty on inputs picked from an ingenerable sequence. Specifically, we show that if a uniform (respectively, exponential-time) circuit family $(C_n)_{n\in\mathbb{N}}$, which takes as input a string of at least super-logarithmic and at most polynomial length, outputs the bit 1 with negligible probability on uniformly random inputs, then it must output 1 with negligible probability on inputs fixed from a computably (respectively, exponential-time) ingenerable sequence $(s_n)_{n\in\mathbb{N}}$. Even more technically, we show that

$$\mathop{\mathbb{E}}_{x} \langle 1| C_n(x) |1\rangle = \eta(n) \implies \langle 1| C_n(s_n) |1\rangle = \eta'(n) \tag{1}$$

for two possibly distinct negligible functions η and η', provided that $(s_n)_{n\in\mathbb{N}}$ is a suitable ingenerable sequence. We emphasize here that this holds for *all* uniform (respectively, exponential-time) circuit families $(C_n)_{n\in\mathbb{N}}$ while the ingenerable sequence $(s_n)_{n\in\mathbb{N}}$ remains fixed and independent of the circuit family.

This result allows us to derandomize certain cryptographic games related to cloning. For example, in Theorem 13, we demonstrate the existence of a fixed sequence of Wiesner states[3] which cannot be cloned by any uniform circuit family. Specifically, we show that any uniform circuit family attempting to copy this fixed sequence will generate states having negligible fidelity with perfect copies of the original ones. This follows from applying our above result to the theorem of Molina, Vidick, and Watrous showing that the ability to clone a uniformly random Wiesner state is negligible in the length of this state [MVW13]. It then suffices to replace the uniformly random Wiesner state with one which is determined by a computably ingenerable sequence.

[3]A Wiesner state, introduced to quantum cryptography in [Wie83], is any tensor product of the single-qubit computational basis states, $|0\rangle$ and $|1\rangle$, or of their Hadamard conjugates, $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$. For two strings x, θ of the same length, we let $|x^\theta\rangle = H^{\theta_1} |x_1\rangle \otimes \cdots \otimes H^{\theta_n} |x_n\rangle$, where H is the single-qubit Hadamard operator, denote a Wiesner state.

1.1.4 A Promise Problem with Uncloneable Advice from MoE games

We now present our first main result: a promise problem $P = (P_{\text{yes}}, P_{\text{no}})$ which admits uncloneable advice. This is the content of Section 5.3. Recall that for a promise problem we do not require P_{yes} and P_{no} to partition $\{0,1\}^*$: some strings may not be an element of either sets, but we are promised to only get problem instances from $P_{\text{yes}} \cup P_{\text{no}}$.

First, we consider an exponential-time ingenerable sequence $(x_n)_{n\in\mathbb{N}}$ and define the set P_{yes} to be all strings y where the inner product $x_n \cdot y$ is 1. Similarly, the set P_{no} will be all strings which have an inner product of 0 with the elements of the ingenerable sequence. Evidently, this problem cannot be efficiently solved without some advice on the sequence $(x_n)_{n\in\mathbb{N}}$. Indeed, if we were able to solve this problem without any advice, we would be able to generate the ingenerable sequence. Now, the obvious advice to directly give is x_n as this would certainly allow an honest user to solve all problem instances. However, this advice is trivially cloneable as it is a classical piece of information. So, instead, we will give as advice the Wiesner state $|x_n^{\theta_n}\rangle$ for a string θ_n which will be determined shortly. We know from prior works on monogamy-of-entanglement (MoE) games [TFKW13] that it is infeasible for an adversary to split a random Wiesner state $\left|x^{\theta}\right\rangle$ in such a way that two parties can recover the string x if they know θ, provided that the initial splitting adversary does not know θ. We recall and formalize this result as Lemma 36 in our detailed exposition. In some sense, the state $\left|x^{\theta}\right\rangle$ encodes information on the string x in an uncloneable way, which is what we need. However, three interrelated issues now arise.

The first problem is that if θ_n is fixed — and also x_n for that matter — how do we make sure that an adversary is not able to create copies of the fixed state $|x_n^{\theta_n}\rangle$? Lemma 36 considers the case of uniformly random Wiesner states, not a fixed sequence of such states. The solutio is to also take θ_n from an exponential-time ingenerable sequence. For technical reasons, which can be roughly understood as making sure that x_n and θ_n are not too correlated, we will consider a single exponential-time ingenerable sequence $(s_n)_{n\in\mathbb{N}}$ and parse each string s_n as the concatenation (x_n, θ_n) of two strings of length n. We can then apply our generic derandomization theorem for ingenerable sequences, Theorem 10 discussed previously in Section 1.1.3, to the MoE game described above to show that no triplet of polynomial-time (or, even, exponential-time) adversaries can have a non-negligible advantage at winning this game when challenged with the fixed state $|x_n^{\theta_n}\rangle$.

Second, we have argued above that polynomial-time adversaries cannot win the MoE game of [TFKW13] with a non-negligible probability when challenged with the fixed states $|x_n^{\theta_n}\rangle$. However, this game requires both guessing adversaries to correctly determine the complete string x_n to win. Solving the problem P only requires them to determine the inner product $x_n \cdot y$ for some string y, a task which may be easier. How can we bridge this gap? It suffices to use the recent result of Kundu and Tan [KT22], formalized in Lemma 1, which states that a negligible probability of both guessing adversaries determining x_n implies a that the probability that both guessing adversaries determining $x_n \cdot y$, for independent and uniformly random values of y, is at most negligibly greater than $\frac{1}{2}$.

Finally, how does the honest user of the advice know θ_n? We have sketched above our argument as to why the advice state $|x_n^{\theta_n}\rangle$ is uncloneable, but not yet as to how it is useful. The naive way to use this state as advice for this problem is to measure it in the θ_n-Wiesner basis to obtain the string x_n. This requires the knowledge of θ_n, something which is ingenerable, and not known to the honest user so far. To overcome this issue, we will modify our problem to be a *promise* problem. The promise is that every problem instance y is prefixed with the fixed string $\theta_{|y|}$. In other words, we are giving θ_n as part of the problem instance. Because of this, θ_n is not accessible to an adversary attempting to split the advice state before knowing a problem instance.

In summary, we start with the monogamy-of-entanglement game of [TFKW13], derandomize it via exponential-time ingenerable sequences, and then apply Kundu and Tan's lemma. We then argue that a promise problem with uncloneable advice can be extracted from this construction.

We further note that this construction can be generalized by replacing the monogamy-of-entanglement game with any sufficiently secure uncloneable encryption scheme [BL20] where the key k_n used and the message m_n to be be encoded play the role of θ_n and x_n, respectively, and are also determined by an ingenerable sequence. We note that the resulting construction, up to the use of ingenerable sequences to derandomize the scheme, is conceptually similar to Kundu and Tan's construction of uncloneable encryption for a single-bit message with *variable keys* [KT22].

As a last remark, we highlight that we use *exponential-time* ingenerable sequences in this construction. As previously mentioned, we show in this work that there exist such sequences which can be computed in triple exponential time. Thus, if P is constructed from an exponential-time ingenerable sequence which can be computed in triple exponential time, then P can be enumerated by a classical deterministic Turing machine. Hence, our result is more than merely existential: we show how to construct P in such a way that the *yes* and *no* instances can be enumerated and we explicitly give an algorithm which does this enumeration.[4]

1.1.5 A Language with Uncloneable Advice from Copy-Protected PRFs

Our second main result is the construction of a *language* with uncloneable advice. Here, we *do* require P_{yes} and P_{no} to partition $\{0,1\}^*$. This is the content of Section 5.4 and Section 5.5

To construct this language, we leverage even more explicitly the connection between advice and programs: Our starting building block is the assumption that it is possible to copy-protect pseudo-random functions with super-logarithmic outputs. This can be achieved under certain assumptions [CLLZ21].

At a high level, we can understand the problem instances of the promise problem described in the previous section as pairs (θ, y) where the string θ was used as an "input" for the "program" encoded by the Wiesner state $\left|x^\theta\right\rangle$. Evaluating a Wiesner state in this paradigm then consists of measuring it in the θ-Wiesner basis and outputting the resulting x. The reason we obtained a promise problem, and not a language, is that this "program" has a deterministic answer on only one input: θ.

Copy-protected programs do not suffer this limitation of only having one "good input". If ρ_f is the copy-protected program of the function f, then evaluating ρ_f on any input x will yield $f(x)$ with high probability. Thus, we can reuse the same template as in the previous promise problem: problem instances are pairs (x, y) where x is treated as an input to a program state ρ_f to obtain a string $f(x)$ with near certainty. We then take the inner product $f(x) \cdot y$ to determine if (x, y) is an element of the language. However, since ρ_f can be properly evaluated on *any* input, unlike the Wiesner state "program", we need not promise that the input x is some particular well-behaved string: it can be arbitrary. Thus, we can obtain a language, and not merely a promise problem from this type of construction.

Here, as for our promise problem, the result of Kundu and Tan (Lemma 1) is used to relate the difficulty of two guessers independently and simultaneously determining $f(x) \cdot y$ to their difficulty of determining $f(x)$. We also make use of exponential-time ingenerable sequences to derandomize the function which is copy-protected all the while maintaining the uncloneability of the resulting program.

[4]Strictly speaking, we give a deterministic classical algorithm in the proof of Theorem 9 computing in triple exponential time the elements of an exponential-time ingenerable sequence. It is then easy to see how to construct an algorithm enumerating P_{yes} and P_{no} when given an algorithm which computes θ_n and x_n for any $n \in \mathbb{N}$.

1.2 Open Questions

This work represents the first attempt to study uncloneable advice and shows that this concept is, in principle, achievable. Beyond identifying other languages and promise problems which admit uncloneable advice, this work leaves many other open questions. We highlight two such questions, mentioned in the previous discussion, which we believe are of particular interest.

Can we say more on the complexity of cloning quantum states? As illustrated in Figure 1 and discussed in Section 1.1.2, our work establishes the existence of sequences of states at the simultaneous extremes of cloning and generating complexity.

Are there scenarios between the extremes sketched above? In other words, how densely populated is the lower-right triangle of Figure 1? Our results imply the existence of a sequence of states that are impossible to clone in polynomial time — or indeed in exponential time — but which can be perfectly generated in triple exponential time. Can we close this gap? For example, can we find a fixed sequence which cannot be cloned in polynomial time, but can be generated in exponential time?

In a sense, our work initiates a theory of "computability of cloning" by showing a separation between sequences of states which can and cannot be cloned. We believe that further and more fine-grained work in this direction would help establish a theory on the complexity of cloning quantum states. We hope that such a theory would then provide useful tools for uncloneable cryptography and other aspects of quantum information theory.

Can we "boost" the success probability of an honest party using uncloneable advice while maintaining the uncloneability property? As discussed in Section 1.1.1, the standard error reduction technique of giving multiple copies of the advice state evidently fails to maintain the required uncloneability property of the initial state. Furthermore, the well-known amplification technique of Marriott and Watrous [MW05], developed in the context of quantum *proofs* and the **QMA** complexity class, does not appear to be directly applicable to the context of uncloneable advice. We highlight two obstacles in applying the result of Marriott and Watrous to uncloneable advice. First, the Marriott-Watrous technique requires changing the proof state and it is unclear if this modification to the state would preserve the required uncloneability property. Second, and perhaps more fatal, is that directly applying the Marriott-Watrous technique to quantum *advice*, instead of quantum *proofs*, does not in general yield quantum *advice* as the required change to the quantum state depends on problem instance for which we wish to increase the probability of success. This dependency is acceptable in the context of *proofs* as these may depend on the problem instance, but is unacceptable in the context of *advice* which must be independent of the problem instance.

Addressing this question would elucidate the relation between the complexity classes **neglQP**/upoly and **BQP**/upoly. Moreover, finding a generic boosting technique for uncloneable advice could have wider repercussions in uncloneable cryptography by effectively lowering the necessary threshold for correctness.

1.3 Acknowledgements

This work was supported by the Air Force Office of Scientific Research under award number FA9550-20-1-0375, Canada's NSERC, and the University of Ottawa's Research Chairs program.

2 Preliminaries

2.1 Notation and Terminology

Sets, bit strings, miscellaneous. We let $\mathbb{N}$ denote the non-negative integers, $\mathbb{R}$ denote the real numbers, $\mathbb{R}^+$ denote the strictly positive real numbers, and $\mathbb{R}_0^+ = \mathbb{R}^+ \cup \{0\}$, denote the non-negative real numbers. For any $n \in \mathbb{N}$, we let $[n]$ denote the set $\{i \in \mathbb{N} \ : \ 1 \leq i \leq n\}$. For any set $\mathcal{X}$, we let $\mathcal{P}(\mathcal{X})$ denote the power set of $\mathcal{X}$.

For all $n \in \mathbb{N}$, let $\{0,1\}^n$ denote the set of bit strings of length n and let $\{0,1\}^* = \cup_{n\in\mathbb{N}}\{0,1\}^n$ denote the set of all bit strings of finite length. We let $|x|$ denote the length of the bit string x and we denote the empty bit string by ε, which is to say that $\{0,1\}^0 = \{\varepsilon\}$. If x and y are two strings of the same length, we let $x \cdot y$ denote their inner product modulo 2. The result of concatenating two bit strings $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$ is denoted by $(x,y) \in \{0,1\}^{n+m}$, using implicitly the isomorphism $\{0,1\}^n \times \{0,1\}^m \cong \{0,1\}^{n+m}$. For all $n \in \mathbb{N}$ we let 0^n and 1^n denote the all-zero and all-one bit strings in $\{0,1\}^n$, respectively.

Logarithms are always taken in base 2.

Probability theory. If $\mathcal{X}$ is a set, we use $\mathbb{E}_{x\leftarrow\mathcal{X}}\, f(x)$ to denote the expectation of f when x is sampled uniformly at random from $\mathcal{X}$. If α is a random variable, we use $x \leftarrow \alpha$ to denote that x is sampled according to α.

Asymptotic analysis. We will use the ω, Ω, and $\mathcal{O}$ notation to characterize the asymptotic behaviour of functions from $\mathbb{N}$ to $\mathbb{R}_0^+$ [Knu76].[5]

Given two functions $f, g : \mathbb{N} \to \mathbb{R}$, we say that f is *eventually smaller* than g if there exists a $n_0 \in \mathbb{N}$ such that $n \geq n_0 \implies f(n) \leq g(n)$. A function $f : \mathbb{N} \to \mathbb{R}$ is said to be *negligible* if it is eventually smaller than $n \mapsto n^{-c}$ for all $c \in \mathbb{N}$. See, for example, [Bel02]. Recall that 2^{-f} is negligible if and only if $f \in \omega(\log)$ and that if η is negligible, f is non-decreasing, and $f \in \Omega(n^r)$ for some $r \in \mathbb{R}^+$, then $\eta \circ f$ is also negligible.

Turing machines and their run times. We only consider deterministic Turing machines in this work and we refer to standard references, such as [AB09] or [LV19], for more details. We do, however, establish some notation and terminology.

A Turing machine $\mathtt{T}$ takes as input a string $x \in \{0,1\}^*$ and either (i) halts and produces as output a string $y \in \{0,1\}^*$, or (ii) never halts, in which case it does not produce an output. We write $\mathtt{T}(x) = y$ to denote the fact that the Turing machine $\mathtt{T}$ halts and produces the output y on input x. Turing machines perform their calculations in a series of discrete steps. We say that a Turing machine $\mathtt{T}$ runs in polynomial-time if and only if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that on input of any string x, the machine $\mathtt{T}$ halts after at most $p(|x|)$ steps. For this paper, we identify efficient (quantum) computations with polynomial-time (quantum) computations. We also say that a Turing machine runs in exponential, double exponential, or triple exponential-time if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that on input of any string x, the machine halts after at most $2^{p(|x|)}$, $2^{2^{p(|x|)}}$, or $2^{2^{2^{p(|x|)}}}$ steps, respectively.

We let $\mathcal{T}$ denote the set of all Turing machines. Note that $\mathcal{T}$ is countable and so there exists a bijective map $\tau : \mathbb{N} \to \mathcal{T}$.

[5]Note that we follow Brassard [Bra85] by always treating $\omega(f)$, $\Omega(f)$, and $\mathcal{O}(g)$ as sets and avoiding "one-way equations".

Finally, for maps $f : \mathbb{N} \to \mathbb{N}$, we say that a Turing machine $\mathtt{T}$ computes f, perhaps within some time bound, if on input of 1^n, i.e. the bit string composed of n instances of 1, it outputs $1^{f(n)}$. In particular, if $\mathtt{T}$ can computes $f : \mathbb{N} \to \mathbb{N}$ and $\mathtt{T}(1^n)$ halts within $T(n)$ steps, then $f(n) \leq T(n)$ as $\mathtt{T}$ can write at most one symbol per step.

Quantum mechanics, Hilbert spaces, operators, and channels. Quantum mechanics, to the extent needed for this work, is a theory concerning linear operators on finite-dimensional complex Hilbert spaces. We refer the reader to the standard introductory textbooks of Watrous [Wat18] and Nielsen and Chuang [NC10] for more details. The set of linear and unitary operators on a given Hilbert space $\mathcal{H}$ are denoted $\mathcal{L}(\mathcal{H})$ and $\mathcal{U}(\mathcal{H})$, respectively. We recall that a *channel* is a completely-positive trace preserving map $\Phi : \mathcal{L}(\mathcal{H}) \to \mathcal{L}(\mathcal{H}')$ and that channels precisely coincide with the set of all possible transformations that a quantum system may undergo over a finite time period. We also recall that by a *state*, we mean a density operator.

In this work, we assume that all Hilbert spaces are finite tensor products of $\mathbb{C}^2$, which is to say that we only consider spaces representing finite numbers of qubits. We suppose that $\left(\mathbb{C}^2\right)^n$ admits $\{|s\rangle\}_{s\in\{0,1\}^n}$ as an orthonormal basis and, when appropriate, we identify a string s with its corresponding density operator $|s\rangle\langle s|$.

We recall that the trace distance between two states is given by $\frac{1}{2}\|\rho - \sigma\|_{\mathrm{Tr}}$ where $\|\cdot\|_{\mathrm{Tr}}$ denotes the trace norm, which is also the $p = 1$ case of the more general Schatten p-norms. We will denote the completely bounded trace norm on channels, also known as the diamond norm, by $\|\cdot\|_{\diamond}$.

Gate sets and quantum circuits. A *gate set* $\mathcal{G}$ is a finite set of channels. We assume that all elements of a gate set are channels of the form $L \mapsto ULU^\dagger$ for a unitary operator $U \in \mathcal{U}(\mathbb{C}^{2^{\otimes n}})$ acting on n qubits whose whose entries in the computational basis are algebraic. Note that n need not be identical across unitaries. There are two possible exceptions: $\mathcal{G}$ may also include the state preparation map $P : \mathbb{C} \to \mathcal{L}(\mathbb{C}^2)$, given by $c \mapsto c\,|0\rangle\langle 0|$, and the single-qubit trace $\mathrm{Tr} : \mathcal{L}(\mathbb{C}^2) \to \mathbb{C}$.

For a gate set $\mathcal{G}$, we let $\langle\mathcal{G}\rangle$ denote the set of channels generated by $\mathcal{G}$. More formally, $\langle\mathcal{G}\rangle$ is precisely the set of all channels Ψ which can be expressed as

$$\Psi = \bigcirc_{i=1}^{d} \bigotimes_{j=1}^{w_i} \Psi_{i,j} \tag{2}$$

for some non-negative integers $d, w_1, \ldots, w_d \in \mathbb{N}$ and channels $\Psi_{i,j} \in \mathcal{G}$. We call an expression C of the form presented on the right-hand side of Equation (2) a *circuit* built from $\mathcal{G}$ implementing the map Ψ and we denote the set of all circuits built from $\mathcal{G}$ by $\langle\mathcal{G}\rangle_{\mathrm{c}}$. Formally, we distinguish between different circuits even if they implement in the same channel but, when convenient, we identify a circuit with the channel it implements. A *circuit family* $C = (C_n)_{n\in\mathbb{N}}$ is a sequence of circuits built from the same gate set. We occasionally write "a circuit family C" without the explicit sequence.

An *encoding* for a gate set $\mathcal{G}$ is a surjective map $e : \{0,1\}^* \to \langle\mathcal{G}\rangle_{\mathrm{c}}$. If $e(s) = C$, we call the string s a description of C. Encodings must satisfy additional conditions [Wat09] which we do not exhaustively list. We require an encoding e to be such that every $\Psi_{i,j}$ term (or lack thereof) in the circuit $e(s)$ must be efficiently computable from i, j, and s. Every gate set admits an encoding.

We say that C is an (a, b)-circuit if it implements a channel $C : \mathcal{L}(\mathbb{C}^{2^{\otimes a}}) \to \mathcal{L}(\mathbb{C}^{2^{\otimes b}})$ for $a, b \in \mathbb{N}$. We say that C is a (a, b)-circuit family if each C_n is an $(a(n), b(n))$-circuit for $a, b : \mathbb{N} \to \mathbb{N}$. We say that a circuit family C is *uniform* if there exists a Turing machine $\mathtt{T}$ which, on input of 1^n, outputs a description of C_n in some prescribed encoding. We say that C is *polynomial-time*, *exponential-time*, or *triple exponential-time* if such a $\mathtt{T}$ exists and runs in polynomial-time, exponential-time, or triple exponential-time, respectively.

A gate set $\mathcal{G}$ is said to be *universal* if for any $\epsilon > 0$ and any channel Φ there exists a $\Psi \in \langle \mathcal{G} \rangle$ such that $\|\Phi - \Psi\|_\diamond \leq \epsilon$. As a consequence of the Solovay-Kitaev theorem [Kit97] and its algorithmic implementations [DN06, BGT21], for any computable map $t : \mathbb{N} \to \mathbb{N}$, any universal gate set $\mathcal{G}$, and any uniform circuit family C' built from any (possibly non-universal) gate set $\mathcal{G}'$, there exists a uniform family C built from $\mathcal{G}$ such that $\|C_n - C'_n\|_\diamond \leq 2^{-t(n)}$. Moreover, if t is efficiently computable, we can replace "uniform" with "polynomial-time" or "exponential-time". Unless otherwise specified, we work with the universal gate set composed of the state preparation map, the single-qubit trace, as well as conjugation by the Toffoli, Hadamard, and phase-shift unitaries [Wat09].

2.2 A Simultaneous Quantum Goldreich-Levin Theorem

We recall a recent result of Kundu and Tan [KT22, Lemma 23] and recast it slightly to consider uniform and polynomial-time circuit families. In some sense, this result extends the quantum Goldreich-Levin theorem [AC02] to a setting with two adversaries.

This lemma concerns a scenario where two non-communicating parties, Bob and Charlie, share a quantum state ρ_{x_B,x_C}. This state represents the information each of these parties has on the value of two bit strings $(x_B, x_C) \in \{0,1\}^n \times \{0,1\}^n$ sampled according to a random variable ξ. The proof given in [KT22] for this lemma shows that if Bob and Charlie can simultaneously, independently, and respectively determine the values of the inner products $x_B \cdot y_B$ and $x_C \cdot y_C$ with probability at least $\frac{1}{2} + \delta$, where the probability is taken over the uniform random sampling of y_B and y_C (which are then given to Bob and Charlie, respectively) and the sampling of (x_B, x_C) from ξ, then they could *instead* simultaneously, independently, and respectively determine the values of x_B and x_C with probability at least $\delta^3/2$.

More technically, the proof shows how to transform a circuit guessing the inner product $x_L \cdot y_L$ for either $L \in \{B, C\}$ into a circuit guessing x_L. The idea is to adapt the Bernstein-Vazirani algorithm [BV97] where the oracle for the inner product map $y \mapsto x \cdot y$ is replaced with a purified version of the circuit guessing the inner product. An important property of this transformation, not explicitly mentioned in [KT22] but which we highlight here, is that this transformation is uniform (respectively, polynomial-time or exponential-time) in the sense that applying it circuit-by-circuit to a uniform (respectively, polynomial-time or exponential-time) circuit family yields another uniform (respectively, polynomial-time or exponential-time) circuit family.

In practice, as in [KT22], we take the contrapositive of the above. We state that if Bob and Charlie have a negligible probability of computing x_B and x_C, then they have at most a negligible advantage in computing $x_B \cdot y_B$ and $x_C \cdot y_C$.

Lemma 1. Let $\ell, b, c : \mathbb{N} \to \mathbb{N}$ be maps and, for all $n \in \mathbb{N}$, let ξ_n be a random variable on the set $\{0,1\}^{\ell(n)} \times \{0,1\}^{\ell(n)}$. For all $n \in \mathbb{N}$ and pairs (x_B, x_C) in the support of ξ_n, let ρ_{n,x_B,x_C} be a density operator on $b(n) + c(n)$ qubits.

If for every pair (B', C') of uniform (b, ℓ)- and (c, ℓ)-circuit families, respectively, the map

$$n \mapsto \mathop{\mathbb{E}}_{(x_B,x_C)\leftarrow\xi_n} \langle x_B, x_C| \left(B'_n \otimes C'_n\right) \left(\rho_{n,x_B,x_C}\right) |x_B, x_C\rangle \tag{3}$$

is negligible, then for every pair (B, C) of uniform $(\ell+b, 1)$- and $(c+\ell, 1)$-circuit families, respectively, the map

$$n \mapsto \mathop{\mathbb{E}}_{\substack{y_B\leftarrow\{0,1\}^{\ell(n)} \\ y_C\leftarrow\{0,1\}^{\ell(n)} \\ (x_B,x_C)\leftarrow\xi_n}} \langle x_B \cdot y_B, x_C \cdot y_C| \left(B_n \otimes C_n\right) \left(y_B \otimes \rho_{n,x_B,x_C} \otimes y_C\right) |x_B \cdot y_B, x_C \cdot y_C\rangle - \frac{1}{2} \tag{4}$$

is negligible.

Moreover, the above holds if we replace all instances of "uniform" with "polynomial-time" or "exponential-time".

We note that an alternative proof of this lemma can also be found in [AKL23].

3 Ingenerable Sequences of Bit Strings

In this section, we define the notion of ingenerable sequences, demonstrate that such sequences exist, and illustrate a few of their simple properties.

Before proceeding, we emphasize that while our definitions and results are formulated in the language of quantum circuits, *there is nothing inherently quantum at play in this section*, with the exception of Section 3.3. In particular, the definitions can easily be rephrased to fit a wide variety of computational models, including classical circuits or classical Turing machines. However, we believe the resulting notions are only interesting when the computational model is *probabilistic* and *uniform*. Moreover, the proofs of our main existence theorems essentially only rely on the assumption that there are countably many instances in the computational model.

Thus, essentially all definitions and results of this section, excluding Section 3.3, still hold if we were to replace the word "circuit", where the "quantum" is implicit, with the words "probabilistic classical Turing machine" or "uniform classical circuits supplied with additional random bits". Our definitions would also hold for "non-uniform circuits", classical or quantum, but our existence theorems would not.

Of course, sequences which are ingenerable with respect to one model of computation need not be ingenerable with respect to another.

3.1 Definitions and Existence

We first introduce terminology concerning sequences of bit strings.

Definition 2. Let $\ell : \mathbb{N} \to \mathbb{N}$ be a map. An ℓ-sequence is a sequence of bit strings $(s_n)_{n\in\mathbb{N}}$ such that $s_n \in \{0,1\}^{\ell(n)}$ for all $n \in \mathbb{N}$.

In practice, we will only concern ourselves with sequences of linear length, such as $2n$-sequences, or of polynomial length. However, the theory developed in this section is applicable to arbitrary ℓ-sequences.

Now, we formally define the notion of q-ingenerability for some map $q : \mathbb{N} \to \mathbb{R}$. In short, a sequence is q-ingenerable if every circuit family under consideration *eventually always fails* to produce the elements of the sequence with probability at least q. We also define a weaker notion of this idea, which we call *weak* ingenerability. A sequence is weakly ingenerable if every circuit family *fails infinitely often* to produce the elements of the sequence with probability at least q.

We immediately see that every q-ingenerable sequence is weakly q-ingenerable. Essentially all of our constructions will use ingenerable sequences, but weak ingenerability will be easier to compare and contrast with other existing definitions and notions.

Finally, we consider ingenerability with respect to two different classes of circuit families. A computably ingenerable sequence is one which cannot be generated by any uniform circuit family. An exponential-time ingenerable sequence is one which cannot be generated by any exponential-time circuit family. Our definition can be adapted in a straightforward way to other classes of circuit families but these two will be sufficient for this work. The main conceptual advantage of exponential-time ingenerable sequences over computably ingenerable sequences is that certain such sequences can actually be computed by Turing machines, unlike computably ingenerable sequences which are, by their natural, uncomputable. See Theorem 9.

Recall from the end of Section 2.1 that, for two maps $a, b : \mathbb{N} \to \mathbb{N}$, an (a, b)-circuit family C is a sequence of circuits $(C_n)_{n\in\mathbb{N}}$ where C_n takes $a(n)$ qubits as input and produces $b(n)$ qubits as output.[6]

Definition 3. Let $\mathcal{G}$ be a gate set and $q : \mathbb{N} \to \mathbb{R}$ be a map. An ℓ-sequence $(s_n)_{n\in\mathbb{N}}$ is computably (respectively, exponential-time) q-ingenerable with respect to $\mathcal{G}$ if for every uniform (respectively, exponential-time) $(0, \ell)$-circuit family C built from $\mathcal{G}$ the inequality

$$\langle s_n| \, C_n(\varepsilon) \, |s_n\rangle < q(n) \tag{5}$$

holds for all but finitely many values of $n \in \mathbb{N}$.

We say that $(s_n)_{n\in\mathbb{N}}$ is *weakly* computably (respectively, exponential-time) q-ingenerable with respect to $\mathcal{G}$ if we only require the above inequality to hold for infinitely many values of $n \in \mathbb{N}$.

It is trivial to show that the concept of ingenerability depends on the underlying gate set $\mathcal{G}$, which is why we make this set explicit in the definition. For example, consider the gate set composed of only the $|0\rangle$ state preparation map, $\mathcal{G} = \{c \mapsto c\,|0\rangle\!\langle 0|\}$, and the gate set $\mathcal{G}' = \mathcal{G} \cup \{L \mapsto XLX\}$ where we add conjugation by the $X = |0\rangle\!\langle 1| + |1\rangle\!\langle 0|$ unitary to $\mathcal{G}$. Clearly, the n-sequence $(1^n)_{n\in\mathbb{N}}$ is computably c-ingenerable with respect to $\mathcal{G}$ for any constant $c > 0$, but not with respect to $\mathcal{G}'$.

The above example is a bit contrived since $\mathcal{G}$ and $\mathcal{G}'$ are not very interesting gate sets. In particular, neither are universal. However, it appears difficult to show that the notion of r-ingenerability coincides even for two distinct but universal gate sets $\mathcal{G}$ and $\mathcal{G}'$. This is due to the fact that universality only guarantees that a circuit built from $\mathcal{G}'$ can *approximate* a circuit built from $\mathcal{G}$ to arbitrarily small *but non-zero* error and that q-ingenerability is defined via a simple inequality. For example, it is *a priori* possible for there to exists an ℓ-sequence $(s_n)_{n\in\mathbb{N}}$ and a uniform $(0, \ell)$-circuit family $(C_n)_{n\in\mathbb{N}}$ built from $\mathcal{G}$ such that $\langle s_n| \, C_n(\varepsilon) \, |s_n\rangle = 1$ for all $n \in \mathbb{N}$, but that for every uniform $(0, \ell)$-circuit family $(C'_n)_{n\in\mathbb{N}}$ built from $\mathcal{G}'$ we would have that $\langle s_n| \, C'_n(\varepsilon) \, |s_n\rangle < 1$ for all $n \in \mathbb{N}$. Such a sequence would be 1-ingenerable with respect to $\mathcal{G}'$, but not with respect to $\mathcal{G}$.

We will later give a slightly modified definition of ingenerability, Definition 6, which will be sufficient for our needs and which will avoid the issue of any possible dependence on the choice of the gate set. This definition will also have the added benefit of not requiring an explicit choice of an q map. First, however, we demonstrate the existence of q-ingenerable ℓ-sequences with respect to any given gate set $\mathcal{G}$ and for any suitable choices of q and ℓ.

Theorem 4. Let $\ell : \mathbb{N} \to \mathbb{N}$ and $q : \mathbb{N} \to \mathbb{R}$ be maps such that $q(n) \cdot 2^{\ell(n)} > n + 1$ for all $n \in \mathbb{N}$. Then, for any gate set $\mathcal{G}$ there exists an ℓ-sequence which is exponential-time q-ingenerable with respect to $\mathcal{G}$.

Proof. We first construct an ℓ-sequence $(s_n)_{n\in\mathbb{N}}$ of bit strings and then we show that it is exponential-time q-ingenerable with respect to $\mathcal{G}$. We interpret the output of all Turing machines in this proof as circuits constructed from $\mathcal{G}$.

Let $\mathcal{R} = \{r_k(n) = 2^{(n+2)^k}\}_{k\in\mathbb{N}}$ be a set of maps. In the context of this proof, these maps will be taken as bounds on the run times of various Turing machines. We note two useful properties of the set $\mathcal{R}$ before proceeding. First, $r_i \leq r_{i+1}$ for all $i \in \mathbb{N}$. Second, if $g \in \mathcal{O}(2^{n^{k_0}})$ for some $k_0 \in \mathbb{N}$, then there exists an $r_{k'} \in \mathcal{R}$ such that $g \leq r_{k'}$.

Now, let $\mathcal{T}$ be the set of all Turing machines and let $\Phi : \mathbb{N} \to \mathcal{T} \times \mathcal{R}$ be a map such that for all $\mathtt{T} \in \mathcal{T}$ there are infinitely many $r \in \mathcal{R}$ such that the pair $(\mathtt{T}, r)$ is in the image of Φ. By the

[6] In this context, we interpret any $k \in \mathbb{N}$ as the constant map $n \mapsto k$.

previously discussed properties of $\mathcal{R}$, this implies that if $\mathtt{T}$ runs in exponential-time, then there exists an $n_{\mathtt{T}}$ such that $\Phi(n_{\mathtt{T}}) = (\mathtt{T}, r)$ and $\mathtt{T}(1^n)$ halts in at most $r(n)$ steps for all $n \in \mathbb{N}$. Such a map Φ exists as $\mathcal{T} \times \mathcal{R}$ is countable and hence there is a surjection from $\mathbb{N}$ to $\mathcal{T} \times \mathcal{R}$. Note that Φ being a surjection is a sufficient, *but not necessary*, condition. To ease later notation, let $\tau : \mathbb{N} \to \mathcal{T}$ and $\rho : \mathbb{N} \to \mathcal{R}$ be the unique maps satisfying $\Phi(n) = (\tau(n), \rho(n))$ for all $n \in \mathbb{N}$.

Now, for every $t, n \in \mathbb{N}$, define the set[7]

$$\mathcal{S}_{t,n} = \begin{cases} \left\{x \in \{0,1\}^{\ell(n)} : \langle x|\, \tau(t)(1^n)(\varepsilon)\, |x\rangle \geq q(n)\right\} & \text{if, on input of } 1^n,\ \tau(t) \text{ halts in} \\ & \text{at most } \rho(t)(n) \text{ steps and yields} \\ & \text{a } (0,\ell(n))\text{-circuit.} \\ \varnothing & \text{else.} \end{cases} \tag{6}$$

In other words, $\mathcal{S}_{t,n}$ is the set of strings generated with probability at least $q(n)$ by the circuit described by the t-th Turing machine on input 1^n, provided that the machine indeed halts in at most $\rho(t)(n)$ steps and outputs the description of a $(0,\ell(n))$-circuit. Under these assumptions, we have that $\sum_{x\in\{0,1\}^{\ell(n)}} \langle x|\, \tau(t)(1^n)(\varepsilon)\, |x\rangle = 1$ which implies that

$$|\mathcal{S}_{t,n}| \leq \frac{1}{q(n)}. \tag{7}$$

Note that this inequality also holds if $\tau(t)(1^n)$ does not halt within the specified number of steps or does not yield a $(0,\ell(n))$-circuit as $|\mathcal{S}_{t,n}| = 0$ in this case and our assumptions imply that q must be strictly positive. Next, define

$$\mathcal{S}_n = \bigcup_{t\in\{0,\dots,n\}} \mathcal{S}_{t,n}. \tag{8}$$

By our assumption on the maps ℓ and q, we have that

$$|\mathcal{S}_n| \leq (n+1)\cdot\frac{1}{q(n)} < 2^{\ell(n)}, \tag{9}$$

which implies that $\{0,1\}^{\ell(n)} \setminus \mathcal{S}_n$ is non-empty. For all $n \in \mathbb{N}$, we take s_n to be the first element, in lexicographic order, of $\{0,1\}^{\ell(n)} \setminus \mathcal{S}_n$. This yields an ℓ-sequence $(s_n)_{n\in\mathbb{N}}$.

Now, we show that $(s_n)_{n\in\mathbb{N}}$ is exponential-time q-ingenerable with respect to $\mathcal{G}$. Consider an exponential-time $(0,\ell)$-circuit family $(C_n)_{n\in\mathbb{N}}$. If no such circuit family exists, then $(s_n)_{n\in\mathbb{N}}$ is vacuously exponential-time q-ingenerable and we are done.

Let $\mathtt{T}$ be an exponential-time Turing machine which generates this circuit family. By our previous discussion on $\mathcal{R}$ and Φ, there exists a $t_{\mathtt{T}} \in \mathbb{N}$ such that $\Phi(t_{\mathtt{T}}) = (\mathtt{T}, r)$ and where $\mathtt{T}(1^n)$ halts in at most $r(n)$ steps. It is now sufficient to show that

$$n \geq t_{\mathtt{T}} \implies \langle s_n|\, C_n(\varepsilon)\, |s_n\rangle < q(n). \tag{10}$$

Assume this was not the case and that there exists an $n' \geq t_{\mathtt{T}}$ such that $\langle s_{n'}|\, C_{n'}(\varepsilon)\, |s_{n'}\rangle \geq q(n')$. Then, by definition, $s_{n'} \in \mathcal{S}_{t_{\mathtt{T}},n'} \subseteq \mathcal{S}_{n'}$. Contradiction, since $s_{n'} \in \{0,1\}^{\ell(n')} \setminus \mathcal{S}_{n'}$. □

[7]Recall that $\tau(t)(1^n)(\varepsilon)$ is a quantum state, when this expression is defined. Indeed, we consider the t-th Turing machine, $\tau(t)$, and run it on input 1^n. This yields the bit string $\tau(t)(1^n)$, assuming the machine halts. Implicitly, we interpret this bit string as a quantum circuit, which itself implicitly defines a quantum channel. Assuming this quantum channel takes as input the empty system, we apply it to the empty quantum state ε to obtain as output the state $\tau(t)(1^n)(\varepsilon)$.

Note that the above proof is constructive in the sense that once q, ℓ, $\mathcal{G}$ and Φ are fixed, then the sequence $(s_n)_{n\in\mathbb{N}}$ is uniquely determined.

We also note that by modifying the set $\mathcal{R}$ used in the proof, we can obtain sequences which are q-ingenerable with respect to circuit families generated by Turing machines operating under various time constraints. An important example for this work is the case where $\mathcal{R} = \{r_k(n) = \infty\}_{k\in\mathbb{N}}$ and where we say that the Turing machine $\mathtt{T}$ halts in at most ∞ steps on a given input if and only if it does halt. For completeness, we formalize this below.

Theorem 5. Let $\ell : \mathbb{N} \to \mathbb{N}$ and $q : \mathbb{N} \to \mathbb{R}$ be maps such that $q(n) \cdot 2^{\ell(n)} > n + 1$ for all $n \in \mathbb{N}$. Then, for any gate set $\mathcal{G}$ there exists an ℓ-sequence which is computably q-ingenerable with respect to $\mathcal{G}$.

Proof sketch. Follow the proof given for Theorem 4, but with $\mathcal{R} = \{r_k(n) = \infty\}_{k\in\mathbb{N}}$. □

We now address the question of the possible dependence of our definition of ingenerable sequences on the choice of the underlying gate set. Looking ahead, this work will be mainly concerned with ℓ-sequences which are computably or exponential-time $(p\cdot 2^{-\ell})$-ingenerable for a polynomial p, but that the particular polynomial is not important. With this in mind, we formulate the following definition of ingenerability which will be sufficient for our needs as well as independent of the choice of the gate set. We also note that this definition naturally extends to the notion of weak ingenerability.

Definition 6. Let $\ell : \mathbb{N} \to \mathbb{N}$ be a map. An ℓ-sequence is said to be computably (respectively, exponential-time) ingenerable if for every gate set $\mathcal{G}$ there exists a polynomial $p_{\mathcal{G}} : \mathbb{N} \to \mathbb{R}$ such that the sequence is computably (respectively, exponential-time) $(p_{\mathcal{G}} \cdot 2^{-\ell})$-ingenerable with respect to $\mathcal{G}$.

By the Solovay-Kitaev theorem, we will see that $(p\cdot 2^{-\ell})$-ingenerability with respect to a universal gate set for any polynomial $p : \mathbb{N} \to \mathbb{R}$ is a sufficient condition to be ingenerable. By definition, it is also a necessary condition. However, a technical condition on the computability of ℓ is needed to formally prove this. The following lemma, which establishes conditions under which ingenerability is a trivial property, will help us account for this technicality.

Lemma 7. Let $\ell : \mathbb{N} \to \mathbb{N}$ be a map. The following two statements are equivalent:

1. Every ℓ-sequence is computably (respectively, exponential-time) ingenerable.
2. Either ℓ is uncomputable (respectively, can't be computed in exponential-time), or $\ell \in \mathcal{O}(\log)$.

Proof. We first show that the second statement implies the first.

If ℓ is uncomputable (respectively, cannot be computed in exponential-time), then there are no uniform (respectively, exponential-time) $(0, \ell)$-circuit families, as a Turing machine producing circuits with the appropriate number of output qubits could be used to compute ℓ. As there are no suitable $(0, \ell)$-circuit family, it is vacuously true that every ℓ-sequence is computably (respectively, exponential-time) ingenerable.

If $\ell \in \mathcal{O}(\log)$, then there exists $n', c \in \mathbb{N}$ such that $n \geq n' \implies \ell(n) \leq c\log(n)$, which implies that $2^{-\ell(n)} \geq n^{-c}$ for all sufficiently large values of n. Now, take $p(n) = n^{c+1}$ and $\tilde{n} = \max\{n', 2\}$ so that $n \geq \tilde{n} \implies p(n) \cdot 2^{-\ell(n)} \geq n \geq 2$. Now, consider any ℓ-sequence $(s_n)_{n\in\mathbb{N}}$ and any uniform $(0, \ell)$-circuit family $(C_n)_{n\in\mathbb{N}}$ built from any gate set $\mathcal{G}$. For all $n \geq \tilde{n}$, we have that

$$\langle s_n|\, C_n(\varepsilon)\, |s_n\rangle \leq 1 < 2 \leq p(n) \cdot 2^{-\ell(n)}. \tag{11}$$

Thus, $(s_n)_{n\in\mathbb{N}}$ is computably ingenerable, which implies that it is also exponential-time ingenerable.

We now show that the first statement implies the second. Assume that every ℓ-sequence is computably (respectively, exponential-time) ingenerable. Further, assume that ℓ is computable (respectively, computable in exponential-time), as otherwise we are done. Consider the gate set $\mathcal{G}$ composed of precisely the $|0\rangle$ state preparation map, the ℓ-sequence $(0^{\ell(n)})_{n\in\mathbb{N}}$, and the uniform (respectively, exponential-time) $(0,\ell)$-circuit family $(C_n)_{n\in\mathbb{N}}$ constructed from $\mathcal{G}$ where C_n is simply the parallel composition of $\ell(n)$ instances of the $|0\rangle$ state preparation map. Clearly, $\left\langle 0^{\ell(n)}\right| C_n(\varepsilon) \left|0^{\ell(n)}\right\rangle = 1$. But, $(0^{\ell(n)})_{n\in\mathbb{N}}$ is assumed to be computably (respectively, exponential-time) ingenerable. Thus, there exists a polynomial $p_{\mathcal{G}} : \mathbb{N} \to \mathbb{R}$ such that $1 < p_{\mathcal{G}}(n) \cdot 2^{-\ell(n)}$ for all sufficiently large values of n. This implies that $\ell(n) \leq \log(p_{\mathcal{G}}(n))$ for all sufficiently large values of n, which is sufficient to conclude that $\ell \in \mathcal{O}(\log)$. □

Lemma 8. Let $\mathcal{G}$ be a universal gate set and $\ell : \mathbb{N} \to \mathbb{N}$ be any map (respectively, any efficiently computable map). If an ℓ-sequence is computably (respectively, exponential-time) $(p\cdot 2^{-\ell})$-ingenerable with respect to $\mathcal{G}$ for a polynomial $p : \mathbb{N} \to \mathbb{R}$, then it is computably (respectively, exponential-time) ingenerable.

Proof. Let $(s_n)_{n\in\mathbb{N}}$ be an ℓ-sequence which is $(p \cdot 2^{-\ell})$-ingenerable with respect to $\mathcal{G}$. Further, assume that ℓ is computable, as otherwise $(s_n)_{n\in\mathbb{N}}$ is already known to be ingenerable by Lemma 7.

Let $\mathcal{G}'$ be any gate set and let $(C'_n)_{n\in\mathbb{N}}$ be a uniform (respectively, exponential-time) family of $(0,\ell)$-circuits built from this set. Now, let $(\tilde{C}_n)_{n\in\mathbb{N}}$ be a uniform family of $(0,\ell)$-circuits with gates from the universal gate set $\mathcal{G}$ such that $\|C_n - \tilde{C}_n\|_\diamond \leq 2^{-\ell(n)}$ for all $n \in \mathbb{N}$. Such a uniform (respectively, exponential-time) family exists by the Solovay-Kitaev theorem. Since $(s_n)_{n\in\mathbb{N}}$ is computably (respectively, exponential-time) $(p \cdot 2^{-\ell})$-ingenerable with respect to $\mathcal{G}$, we have that

$$\langle s_n| C_n(\varepsilon) |s_n\rangle \leq \langle s_n| \tilde{C}_n(\varepsilon) |s_n\rangle + 2^{-\ell(n)} < (p(n)+1)\cdot 2^{-\ell(n)} \tag{12}$$

for all sufficiently large values of n. Hence, the ℓ-sequence $(s_n)_{n\in\mathbb{N}}$ is computably (respectively, exponential-time) $(p+1)\cdot 2^{-\ell}$-ingenerable with respect to $\mathcal{G}'$. This yields the desired result as $p+1$ is a polynomial. □

The culmination of the results in this section is the following theorem which asserts the existence of computably ingenerable and exponential-time ℓ-sequences for any suitable choice of ℓ.

Theorem 9. There exists a computably ingenerable ℓ-sequence for any map $\ell : \mathbb{N} \to \mathbb{N}$.

Moreover, if ℓ is efficiently computable, then there exists an exponential-time ingenerable ℓ-sequence which can be computed in triple exponential-time by a classical deterministic Turing machine.

Proof. Let $\mathcal{G}$ be a universal gate set and let $p(n) = n + 2$. Then, by Theorem 5, there exists a computably $(p \cdot 2^{-\ell})$-ingenerable ℓ-sequence with respect to $\mathcal{G}$. By Lemma 8, this is a computably ingenerable ℓ-sequence.

Keeping the same $\mathcal{G}$ and p and assuming that ℓ can be efficiently computed, we now show how to fix the map Φ in the proof of Theorem 4 to obtain an ℓ-sequence which is exponential-time ingenerable with respect to $\mathcal{G}$ but which can be computed in triple exponential-time by a classical deterministic Turing machine. By Lemma 8, this is an exponential-time ingenerable sequence.

Recall that $\mathcal{R} = \left\{ r_k(n) = 2^{(n+2)^k} \right\}_{k\in\mathbb{N}}$. Let $\langle\cdot\rangle : \{0,1\}^* \to \mathcal{T}$ be a mapping from bit strings to Turing machines and let $\mathtt{U}$ be a Turing machine satisfying

$$\mathtt{U}(s, x, 1^t) = \begin{cases} (1, \langle s\rangle(x)) & \text{if } \langle s\rangle \text{ halts within t steps on input } x \\ 0 & \text{else} \end{cases} \tag{13}$$

and which runs in polynomial time.[8] In other words, $\mathtt{U}$ is an efficient universal Turing machine with a time bound.[9] Further, let $b : \mathbb{N} \to \{0,1\}^*$ be the natural bijection between the non-negative integers and the list of all finite bit strings taken in lexicographic order, starting with the empty string, i.e. $b(0) = \varepsilon$, $b(1) = 0$, $b(2) = 1$, and $b(3) = 00$.

Let $(\alpha_n)_{n\in\mathbb{N}}$ and $(\beta_n)_{n\in\mathbb{N}}$ be two sequence of non-negative integers such that the following three conditions are satisfied. First, $\alpha_n, \beta_n \leq n$ for all $n \in \mathbb{N}$. Second, $n \mapsto (\alpha_n, \beta_n)$ can be computed efficiently in n. Third, for every $a \in \mathbb{N}$, there are infinitely many distinct $b \in \mathbb{N}$ such that (a, b) is in the image of $n \mapsto (\alpha_n, \beta_n)$.[10] We define Φ by $n \mapsto (\langle b(\alpha_n)\rangle, r_{\beta_n})$.

Note that by our assumptions on $(\alpha_n)_{n\in\mathbb{N}}$ and $(\beta_n)_{n\in\mathbb{N}}$, the map Φ we have constructed here satisfies the assumptions made of it in the proof of Theorem 4, namely that for every $\mathtt{T} \in \mathcal{T}$ there are infinitely many $r \in \mathcal{R}$ such that $(\mathtt{T}, r)$ is in the image of Φ.

Now that we have fixed Φ, the ℓ-sequence $(s_n)_{n\in\mathbb{N}}$ constructed in the proof of Theorem 4 is fixed. It now suffices to show how it can be computed in triple exponential-time.

Recall from the proof of Theorem 4 that s_n is the first element, in lexicographic order, of the non-empty set $\{0,1\}^{\ell(n)} \setminus \mathcal{S}_n$ where $\mathcal{S}_n = \cup_{t=0}^{n} \mathcal{S}_{t,n}$ and

$$\mathcal{S}_{t,n} = \begin{cases} \left\{x \in \{0,1\}^{\ell(n)} : \langle x|\, \tau(t)(1^n)(\varepsilon)\, |x\rangle \geq q(n)\right\} & \text{if, on input of } 1^n,\ \tau(t) \text{ halts in at most } \rho(t)(n) \text{ steps and yields a } (0,\ell(n))\text{-circuit.} \\ \varnothing & \text{else.} \end{cases} \tag{14}$$

where τ and ρ are the projections on the first and second component of Φ, respectively. Thus, to compute s_n, it suffices to be able to check the membership of a string in the sets $\mathcal{S}_{t,n}$ for all $t \leq n$. We describe a Turing machine $\mathtt{T}'$ running in triple exponential-time which finds the first string, in lexicographic order, which is not in any of these sets.

At a high-level, the triple exponential run time of $\mathtt{T}'$ can be explained as follows. The first exponential is due to the need to simulated other Turing machine running in exponential-time. The second exponential is essentially due to the fact that $\mathtt{T}'$ will be simulating more and more complex Turing machines as n grows (specifically due to the fact that $n \mapsto r_n(n)$ is not exponential in n, but is double exponential). The third exponential is due to the "brute force" numerical computation to analyse the quantum circuits produced by the simulated Turing machines.

Formally, let $\mathtt{T}'$ be a Turing machine which implements the following algorithm on input of 1^n:

1. Compute $\ell(n)$.

   (By assumption, this can be done efficiently.)

[8]Technically, $\mathtt{U}$ should receive $(r(s), 01, r(x), 01, 1^t)$ as input, where $r : \{0,1\}^* \to \{0,1\}^*$ is the map which repeats each element of the argument twice, i.e. $r(010) = 001100$. This, with the help of the separators 01, allows $\mathtt{U}$ to unambiguously parse and separate s, x, and 1^t. However, for simplicity, we omit these extra details in this proof.

[9]Such a machine $\mathtt{U}$ and encoding $\langle\cdot\rangle$ can be found implicitly in [NW06]. In particular, Theorem 7 of [NW06] shows that the run time of their universal Turing machine $U_{3,11}$ is polynomial in the run time of the simulated machine M and the number of states of M. Adding a mechanism to count the number of simulated steps can be done with at most a polynomial overhead. Moreover, they implicitly provide an injective encoding $e : \mathcal{T} \to \{0,1\}^*$ by, essentially, simply listing all transition rules of the Turing machine in a prescribed manner. From this encoding e, we can define the surjection $\langle\cdot\rangle : \mathcal{T} \to \{0,1\}$ by $\langle s\rangle = e^{-1}(s)$ if s is in the image of e, and where $\langle s\rangle$ is otherwise the 2 state Turing machine which immediately and always transitions from its initial state to its halting state in the first step. Note that this implies that the number of states of $\langle s\rangle$ is at most $|s| + 2$. As a final note, the Turing machines of [NW06] use a non-binary alphabet. However, this can be transformed to a Turing machine using the alphabet $\{0,1\}$ with a constant multiplicative overhead in time and space usage using standard techniques [AB09].

[10]As a concrete example, we can take $(\alpha_n)_{n\in\mathbb{N}}$ to be the sequence obtained by concatenating each finite sequence of the form $0, 1, 2, \ldots, n$ for all $n \in \mathbb{N}$ and take $(\beta_n)_{n\in\mathbb{N}}$ to be the sequence obtained by repeating, in increasing order, every element $n \in \mathbb{N}$ exactly $n + 1$ times.

2. Initialize a string $s = 0^{\ell(n)}$ and a counter $t = 0$, both of which will be updated as the algorithm proceeds.

   (This can be done efficiently in $\ell(n)$.)

3. If $t > n$, go to step 4. Else, do the following:

   (a) Compute α_t and β_t.
   (This can be done efficiently in t, and so efficiently in n as $t \leq n$.)

   (b) Run $\texttt{U}(b(\alpha_t), 1^n, 1^{r_{\beta_t}(n)})$ and let z denote the output.
   (By assumption, there exists a polynomial p, which we can assume to be non-decreasing, such that this takes at most $p(|b(\alpha_t)| + n + r_{\beta_t}(n))$ steps. As $\alpha_t, \beta_t \leq t \leq n$, $|b(n)| \leq n$, and $r_i \leq r_{i+1}$, this then takes at most $p(2n + r_n(n))$ steps. Note that $r_n(n) = 2^{(n+2)^n}$. Since $(n+2)^n \in \mathcal{O}(2^{n^2})$, we conclude that $r_n(n)$ and $p(2n + r_n(n))$ are at most double exponential in n.)

   (c) If the first bit of z is 0, which is to say that the Turing machine being simulated did not terminate within the desired number of steps, do nothing. Else, let z' be all but the first bit of z and check that z' encodes a $(0, \ell(n))$ circuit. If it does not, do nothing. If it does, then do the following:
   (This check can be done efficiently in $\ell(n)$ and the length of z'. Since $\ell(n)$ is at at most double exponential in n and the length of z' is at most double exponential in n, as it is the result of a simulation which ran for at most $r_n(n)$ simulated steps, then this check can be executed in time double exponential in n.)

      i. Let C be the $(0, \ell(n))$ circuit described by z'. Check if $\langle s| C(\varepsilon) |s\rangle < p(n) \cdot 2^{-\ell(n)}$. If so, increment t by 1 and go back to step 3. If not, update s to be the next string of length $\ell(n)$, in lexicographic order, and execute this step again.
      (The time complexity of this check is dominated by the complexity of computing the value $\langle s| C(\varepsilon) |s\rangle$. Recall that C is described by z' which was produced after at most $r_n(n)$ simulated steps. Hence, $|z'| \leq r_n(n)$ and so C can have at most $r_n(n)$ gates. Thus, we can compute $\langle s| C(\varepsilon) |s\rangle$ in time exponential in $r_n(n)$, which is triple exponential in n as $r_n(n)$ is double exponential in n.)

4. Output s.

The fact that $\texttt{T}'(1^n)$ indeed halts and outputs s_n follows from the proof of Theorem 4. The time complexity is dominated by step 3.c.i, with each execution of that step taking time which is at most triple exponential in n. Furthermore, this step is executed at most $(n+1)2^{\ell(n)}$ times, which is itself at most triple exponential in n. We conclude that $\texttt{T}'$ runs in at most triple exponential time on input of 1^n, which is the desired result. □

In Appendix A, we study some links between computably ingenerable sequences and Martin-Löf random sequences. Recall that a Martin-Löf sequence is, essentially, a fixed and infinite sequence of bits $w \in \{0,1\}^\infty$ which passes all possible computable tests of randomness. We show in this appendix that every Martin-Löf random sequence yields a weakly computably ingenerable ℓ-sequence following the application of a natural bijection and that there exists computably ingenerable ℓ-sequences which are not Martin-Löf random under the action of the same bijection. We also show the existence of weakly computably ingenerable sequences which are not computably ingenerable.

3.2 Replacing Uniformly Random Strings with Ingenerable Strings

The goal of this section is to prove Theorem 10 which is stated below. This theorem can be understood as stating that if a given computation succeeds with sufficiently small probability over uniformly random inputs, than applying that same computation on fixed inputs determined by an ingenerable ℓ-sequence also yields a negligible success probability, provided that ℓ is large enough.

Theorem 10. Let $(s_n)_{n\in\mathbb{N}}$ be a computably (respectively, exponential-time) ingenerable ℓ-sequence for any $\ell : \mathbb{N} \to \mathbb{N}$ (respectively, $\ell \in \mathcal{O}(n^k)$ for some $k \in \mathbb{N}$). Then for any uniform (respectively, exponential-time) family $(C_n)_{n\in\mathbb{N}}$ of $(\ell, 1)$-circuits, there exists a polynomial p such that for all sufficiently large values of n (where "sufficiently large" might depend on $(C_n)_{n\in\mathbb{N}}$) we have that

$$\langle 1| C_n(s_n) |1\rangle \leq p(n) \cdot \left(2^{-\ell(n)} + \mathop{\mathbb{E}}_{x\leftarrow\{0,1\}^{\ell(n)}} \langle 1| C_n(x) |1\rangle\right). \tag{15}$$

In particular, if $\ell \in \omega(\log)$ and $n \mapsto \mathbb{E}_{x\leftarrow\{0,1\}^{\ell(n)}} \langle 1| C_n(x) |1\rangle$ is negligible, then so is $n \mapsto \langle 1| C_n(s_n) |1\rangle$.

The proof of Theorem 10 is simply an application of the following lemma which considers a naive and non-efficient way to transform a circuit C for a decision problem into a circuit $\tilde{C}$ for a search problem. It then gives a lower bound on the probability that $\tilde{C}$ outputs any given string.

Lemma 11. Let $n \in \mathbb{N}$ and let C be an $(n, 1)$-circuit. Now, let $\tilde{C}$ be a $(0, n)$-circuit which implements the following algorithm:

1. Initialize an empty set $\mathcal{S}$.
2. Iterating over all strings $x \in \{0,1\}^n$, do the following: Run $C(x)$ and measure the output in the computational basis. If the result is 1, add x to $\mathcal{S}$. Else, do nothing.
3. Sample uniformly at random a string x from $\mathcal{S}$ and output it. If $\mathcal{S}$ is empty, sample $\tilde{x}$ uniformly at random from $\{0,1\}^n$.

Then, for any $s \in \{0,1\}^n$, we have that

$$\frac{\langle 1| C(s) |1\rangle}{1 + \sum_{x\in\{0,1\}^n\setminus\{s\}} \langle 1| C(x) |1\rangle} \leq \langle s| \tilde{C}(\varepsilon) |s\rangle. \tag{16}$$

We give the proof of the above lemma in Appendix B.1. The proof simply neglects the case where $\mathcal{S}$ is empty by the time the algorithm executes step 3 and notes that the left-hand side of the above inequality is essentially $\Pr[s \text{ is outputted.} \mid s \text{ is in } \mathcal{S}.] \cdot \Pr[s \text{ in } \mathcal{S}]$, up to an application of Jensen's lemma.

We can now prove Theorem 10.

Proof of Theorem 10. Let $(\tilde{C}_n)_{n\in\mathbb{N}}$ be a $(0, \ell)$-circuit family such that $\tilde{C}_n$ implements the algorithm described in Lemma 11 with respect to C_n for all $n \in \mathbb{N}$.

Note that if $(C_n)_{n\in\mathbb{N}}$ is an exponential-time family and $\ell \in \mathcal{O}(n^k)$, then $(\tilde{C}_n)_{n\in\mathbb{N}}$ is an exponential-time circuit family. Otherwise, if $(C_n)_{n\in\mathbb{N}}$ is a uniform family then so is $(\tilde{C}_n)_{n\in\mathbb{N}}$.

We then have that

$$\langle 1| C_n(s_n) |1\rangle \leq \langle s_n| \tilde{C}_n(\varepsilon) |s_n\rangle \left(1 + \sum_{x\in\{0,1\}^{\ell(n)}\setminus\{s_n\}} \langle 1| C_n(x) |1\rangle\right) \tag{17}$$

for all $n \in \mathbb{N}$. As $(s_n)_{n\in\mathbb{N}}$ is a computably (respectively, exponential-time) ingenerable ℓ-sequence, there exists a polynomial p such that, for all sufficiently large values of n, we have that $\langle s_n| \tilde{C}_n(\varepsilon) |s_n\rangle \leq p(n) \cdot 2^{-\ell(n)}$. Thus,

$$\langle 1| C_n(s_n) |1\rangle \leq \frac{p(n)}{2^{\ell(n)}} \left(1 + \sum_{x\in\{0,1\}^{\ell(n)}\setminus\{s_n\}} \langle 1| C_n(x) |1\rangle \right) \tag{18}$$

for all sufficiently large values of n. Adding the $x = s_n$ term to the sum and distributing the $2^{-\ell(n)}$ factor yields the first desired result.

Furthermore, if $\ell \in \omega(\log)$, then $2^{-\ell}$ is negligible. If $\mathbb{E}_{x\leftarrow\{0,1\}^{\ell(n)}} \langle x| C(x) |x\rangle$ is also negligible, then the upper bound is a polynomial times the sum of two negligible functions and thus is itself negligible. □

3.3 Derandomizing a Cloning Task via Ingenerable Sequences

In this section, we prove that there exists a sequence of fixed states $(|\psi_n\rangle\langle\psi_n|)_{n\in\mathbb{N}}$ which cannot be cloned by any uniform family of quantum circuits $(C_n)_{n\in\mathbb{N}}$. This will be a simple demonstration of the ideas we will be using in our constructions of uncloneable advice in Section 5.

The following lemma was first shown by Molina, Vidick, and Watrous [MVW13] to formalize the proof of security of Wiesner's quantum money scheme [Wie83]. It can be interpreted as a bound on how well a party can clone a single Wiesner state $|x^\theta\rangle$ selected uniformly at random.

Lemma 12. Let $n \in \mathbb{N}$ and let C be an $(n, 2n)$-circuit. Then,

$$\mathop{\mathbb{E}}_{\substack{x\leftarrow\{0,1\}^n \\ \theta\leftarrow\{0,1\}^n}} \langle x^\theta| \langle x^\theta| C\left(|x^\theta\rangle\langle x^\theta|\right) |x^\theta\rangle |x^\theta\rangle \leq \left(\frac{3}{4}\right)^n. \tag{19}$$

Note that the lemma holds for all quantum channels, but we restrict ourselves to circuits to maintain consistency with the rest of this work.

We can now "derandomize" Lemma 12 by replacing the uniformly random Wiesner state $|x^\theta\rangle$ with a fixed one chosen according to an ingenerable sequence. It then suffices to invoke Theorem 10 to obtain the desired result.

Theorem 13. Let $(s_n)_{n\in\mathbb{N}}$ be a computably ingenerable $2n$-sequence and parse each s_n as a pair of bit strings of length n, *i.e.*: $s_n = (x_n, \theta_n)$ for $x_n, \theta_n \in \{0,1\}^n$ for all $n \in \mathbb{N}$. Then, for any uniform family of $(n, 2n)$-circuits $(C_n)_{n\in\mathbb{N}}$, we have that

$$n \mapsto \langle x_n^{\theta_n}| \langle x_n^{\theta_n}| C_n\left(|x_n^{\theta_n}\rangle\langle x_n^{\theta_n}|\right) |x_n^{\theta_n}\rangle |x_n^{\theta_n}\rangle \tag{20}$$

is a negligible function.

Proof. Conceptually, it suffices to apply Theorem 10 to Lemma 12. However, a few technical details are needed to frame Lemma 12 in a way where Theorem 10 is applicable.

Let $(S_n)_{n\in\mathbb{N}}$ be a uniform family of $(2n, 3n)$-circuits such that on input of $x \otimes \theta$ they output the state $x \otimes \theta \otimes |x^n\rangle\langle x^n|$ for any $x, \theta \in \{0,1\}^n$. Let $(R_n)_{n\in\mathbb{N}}$ be a uniform family of $(4n, 1)$-circuits which, on input of $x \otimes \theta \otimes \rho$ for any strings $x, \theta \in \{0,1\}^n$ and any state ρ on $2n$ qubits, applies the unitary $H^\theta \otimes H^\theta$ to ρ, measures the resulting qubits in the computational basis, and outputs 1 if and only if two copies of x are obtained. It outputs 0 otherwise.

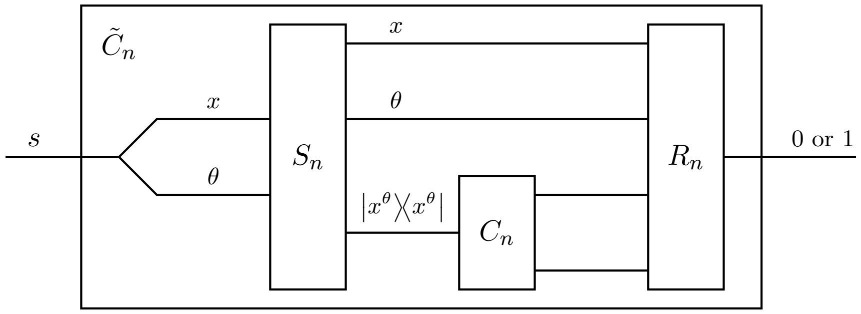


Figure 2: A schematic representation of the $\tilde{C}_n$ circuits constructed in the proof of Theorem 13. The wires are labelled, when possible, with the states they are expected to carry in the context of the proof. Every wire represents n qubits, except the initial and final wires which represent $2n$ and 1 qubits, respectively.

Now, consider the uniform family of $(2n, 1)$-circuits $(\tilde{C}_n)_{n\in\mathbb{N}}$ where $\tilde{C}_n$ is obtained by composing the S_n, C_n, and R_n circuits, in that order and where C_n act on the last n qubits produced by S_n. This is illustrated in Figure 2. At the level of channels, we see that

$$\tilde{C}_n = R_n \circ (\mathrm{Id}_n \otimes \mathrm{Id}_n \otimes C_n) \circ S_n. \tag{21}$$

In particular, note that

$$\langle 1|\, \tilde{C}_n(x \otimes \theta)\, |1\rangle = \langle x^\theta|\, \langle x^\theta|\, C\left(|x^\theta\rangle\langle x^\theta|\right)\, |x^\theta\rangle\, |x^\theta\rangle \tag{22}$$

for all $x, \theta \in \{0,1\}^n$. Thus, by Lemma 12,

$$n \mapsto \mathop{\mathbb{E}}_{\substack{x \leftarrow \{0,1\}^n \\ \theta \leftarrow \{0,1\}^n}} \langle 1|\, \tilde{C}_n(x \otimes \theta)\, |1\rangle \tag{23}$$

is negligible. Hence, by Theorem 10 and the fact that $(s_n)_{n\in\mathbb{N}}$ is computably ingenerable,

$$n \mapsto \langle 1|\, \tilde{C}_n(x_n \otimes \theta_n)\, |1\rangle \tag{24}$$

is also negligible, which, by Equation (22), is the desired result. □

4 State Complexity Classes and their Generalizations to Cloning

In this section, we define a notion of cloning complexity class by generalizing the notion of a state complexity class, as studied by Metger, Rosenthal, and Yuen [RY21, MY23]. Roughly speaking, these works define and study the complexity of generating a given sequence $(\rho_n)_{n\in\mathbb{N}}$ of quantum states. The main result of these works is showing that **stateQIP** = **statePSPACE**, which is to say that the sequences of states which can be efficiently generated with the help of a possibly malicious prover are precisely those which can be generated by polynomial-space quantum circuits. This is a state synthesis analogue to the celebrated result of **QIP** = **PSPACE** [JJUW11].

From our perspective, these prior works concerns the difficulty of creating one copy of ρ_n starting from no copies; we propose that it is natural to also study the difficulty of creating two copies from one, or more generally, b copies starting from a copies for $a > b, a, b \in \mathbb{N}$.[11]

[11] Recent work has also studied the quantum *Kolmogorov* complexity of cloning a quantum state [LFM+23]. The authors obtain results showing that, from this perspective, cloning a quantum state is almost always essentially as hard as creating it.

Definition 14. Let $\mathcal{C}$ be a class of circuit families, $\delta : \mathbb{N} \to \mathbb{R}$ be a function, and $a, b \in \mathbb{N}$ be two non-negative integers. A sequence of states $(\rho_n)_{n\in\mathbb{N}}$ is in the state complexity class $\mathbf{state}_\delta^{a\to b}(\mathcal{C})$ if there exists a circuit family $C \in \mathcal{C}$ and an $n' \in \mathbb{N}$ such that

$$n \geq n' \implies \frac{1}{2}\left\|\rho_n^{\otimes b} - C_n(\rho_n^{\otimes a})\right\|_{\mathrm{Tr}} \leq \delta(n). \tag{25}$$

We also define

$$\mathbf{state}^{a\to b}(\mathcal{C}) = \bigcap_{k\in\mathbb{N}} \mathbf{state}_{n^{-k}}^{a\to b}(\mathcal{C}). \tag{26}$$

If $a = 0$ and $b = 1$, we may omit the $a \to b$ superscript from all the above notation.

Note that in full generality, we could also have a and b in the definition be maps from $\mathbb{N}$ to $\mathbb{N}$ but this will not be necessary for our observations.

Naturally occurring classes of circuit families which may be of interest include the class of uniform circuit families, which we denote $\mathcal{C}_{\mathrm{UNIF}}$, and the class of polynomial-time circuit families, which we denote $\mathcal{C}_{\mathrm{POLY}}$. Letting $\mathcal{C}_{\mathrm{PSPACE}}$ denote the class of "space-uniform circuits" as given in [MY23, Definition 2.2] and **statePSPACE** be as given in [MY23, Definition 2.4], we find, as expected, that $\mathbf{state}(\mathcal{C}_{\mathrm{PSPACE}}) = \mathbf{statePSPACE}$.[12] We note that finding a class of circuit families, tentatively denoted $\mathcal{C}_{\mathrm{QIP}}$, such that $\mathbf{state}(\mathcal{C}_{\mathrm{QIP}}) = \mathbf{stateQIP}$, where the latter is as given in [MY23, Definition 5.1], appears to be a bit more involved and non-trivial. In short, it is unclear how to encode the required soundness property of **stateQIP** into a class of circuit families. Nonetheless, we believe Definition 14 to be a natural generalization of state complexity classes in a wide range of interesting settings.

One interesting result of formalizing cloning complexity classes is that it allows us to make more precise statements about the relations between the complexity of generating and of cloning quantum states. Specifically, we can now state and prove the three lemmas which were sketched in Figure 1 from Section 1.1.2.

The first lemma formalizes the idea that, for essentially all classes of circuit families, cloning a state cannot be more difficult than generating it. For technical reasons, this lemmas does not apply to certain pathological classes of circuit families (*e.g.*: those without the ability to apply the identity channel), or those representing extremely limited computational power (*e.g.*: those with a hard bound on the number of qubits they can process). The idea is that if one is able to generate a sequence of states, then one way to clone the same sequence is to ignore the input and simply generate another copy.

Lemma 15. Let $\mathcal{C}$ be a class of circuit families satisfying the following criteria:

- The class $\mathcal{C}$ can implement the identity on qubits it could generate. More precisely, if $\mathcal{C}$ includes a $(0, \ell)$-circuit family C for some $\ell : \mathbb{N} \to \mathbb{N}$, then $\mathcal{C}$ also includes an (ℓ, ℓ)-circuit family C^{id} where every circuit in this family implements the identity channel.
- The class $\mathcal{C}$ is closed under tensor products. More precisely, given two circuit families $(C_n)_{n\in\mathbb{N}}$ and $(C'_n)_{n\in\mathbb{N}}$ in $\mathcal{C}$, then the family $(C_n \otimes C'_n)_{n\in\mathbb{N}}$ is in $\mathcal{C}$.

Then, for all $\delta : \mathbb{N} \to \mathbb{R}$ and $a, b \in \mathbb{N}$ we have that

$$\mathbf{state}_\delta^{0\to 1}(\mathcal{C}) \subseteq \mathbf{state}_\delta^{1\to 2}(\mathcal{C}). \tag{27}$$

[12]Note that [RY21, MY23] also require that a sequence $(\rho_n)_{n\in\mathbb{N}}$ in **statePSPACE** also satisfies the constraint that each ρ_n is on precisely n qubits. However, they state that this is merely for convenience. We drop this requirement from our definition.

Proof. Let $(\rho_n)_{n\in\mathbb{N}} \in \mathbf{state}_\delta^{0\to 1}(\mathcal{C})$. Let $C \in \mathcal{C}$ and $n' \in \mathbb{N}$ be such that

$$n \geq n' \implies \frac{1}{2}\|\rho_n - C_n(\varepsilon)\|_{\mathrm{Tr}} \leq \delta(n). \tag{28}$$

Consider now the family $C' = (C_n^{\mathrm{id}} \otimes C_n)_{n\in\mathbb{N}}$ which, by our assumptions, is in $\mathcal{C}$. For all $n \geq n'$, we have that

$$\frac{1}{2}\left\|\rho_n^{\otimes 2} - C'_n(\rho_n)\right\|_{\mathrm{Tr}} = \frac{1}{2}\|\rho_n \otimes \rho_n - \rho_n \otimes C_n(\varepsilon)\|_{\mathrm{Tr}} \leq \frac{1}{2}\|\rho_n - C_n(\varepsilon)\|_{\mathrm{Tr}} \leq \delta(n) \tag{29}$$

and so $(\rho_n)_{n\in\mathbb{N}} \in \mathbf{state}_\delta^{1\to 2}(\mathcal{C})$. ☐

The second lemma formalizes the idea that there exists sequences of states which cannot be generated but which can be cloned, even with less computational power. It suffices here to consider sequences of classical information which cannot be consistently produced with high probability. Weakly computably ingenerable sequences satisfy these properties.

Lemma 16. Let $(s_n)_{n\in\mathbb{N}}$ be a weakly computably ingenerable n-sequence. Then, the sequence $(|s_n\rangle\langle s_n|)_{n\in\mathbb{N}}$ is in $\mathbf{state}^{1\to 2}(\mathcal{C}_{\mathrm{POLY}})$ but not in $\mathbf{state}(\mathcal{C}_{\mathrm{UNIF}})$.

Proof. By definition, $(|s_n\rangle\langle s_n|)_{n\in\mathbb{N}}$ is not in $\mathbf{state}(\mathcal{C}_{\mathrm{UNIF}})$ since every uniform circuit family will produce the string s_n with negligible probability for infinitely many values of n.

On the other hand, let $(C_n)_{n\in\mathbb{N}}$ be an $(n, 2n)$-circuit family where C_n simply appends n qubits in the $|0\rangle$ state and then applies a CNOT gate on the $n+k$ qubit conditioned on the k qubit for all $k \in \{1, ..., n\}$. We see that $C_n(|s_n\rangle\langle s_n|) = |s_n\rangle\langle s_n| \otimes |s_n\rangle\langle s_n|$. Since $(C_n)_{n\in\mathbb{N}} \in \mathcal{C}_{\mathrm{POLY}}$, we have that $(|s_n\rangle\langle s_n|)_{n\in\mathbb{N}} \in \mathbf{state}^{1\to 2}(\mathcal{C}_{\mathrm{POLY}})$. ☐

The third and final lemma states that there are sequences of states which simply cannot be cloned by uniform circuit families. It is obtained as a direct corollary from Theorem 13 and can be interpreted as a cloning complexity analogue to the existence of uncomputable functions.

Lemma 17. Let $(s_n)_{n\in\mathbb{N}}$ be a computably ingenerable $2n$-sequence where we parse each s_n as (x_n, θ_n) for two strings $x_n, \theta_n \in \{0,1\}^n$. Then, $(\,|x_n^{\theta_n}\rangle\langle x_n^{\theta_n}|)_{n\in\mathbb{N}} \notin \mathbf{state}^{1\to 2}(\mathcal{C}_{\mathrm{UNIF}})$. By Lemma 15, this also implies that the sequence is not in $\mathbf{state}(\mathcal{C}_{\mathrm{UNIF}})$.

Proof. This is a direct corollary of Theorem 13. ☐

Note that the corollary is a strictly weaker statement than the theorem. Indeed, the theorem demonstrates that it is impossible to clone with *non-negligible* fidelity, via a uniform family of circuits, the Wiesner states described by a computably ingenerable sequence. The corollary only states that it is impossible to clone these states with an *overwhelming* fidelity. We emphasize that our results in Section 5 concerning uncloneable quantum advice will require the stronger guarantee of the form given by the theorem.

5 Uncloneable Advice

The main results of this section are to give a definition for the complexity class of problems which can be solved by polynomial-time quantum computations with negligible errors in the presence of advice that is uncloneable and to give examples of problems in this class. We denote this complexity class **neglQP**/upoly.

As we briefly discussed in Section 1.1.1, we do not denote this class **BQP**/upoly; this is because we lack a generic error reduction technique which can reduce bounded errors to negligible errors, all the while maintaining the uncloneability property of the advice states. In other words, the current state-of-the-art is that it is possible that **neglQP**/upoly $\subseteq$ **BQP**/upoly is a strict containment.

Section overview. This section is organized as follows. We begin by reviewing the basic notions of quantum copy-protection in Section 5.1. In Section 5.2, we formally define the notion of uncloneable advice and give some basic remarks on this definition. We also emphasize some parallels between quantum copy-protection and uncloneable advice. In Section 5.3, we describe a promise problem which unconditionally admits uncloneable advice. In Section 5.4, we describe a language with uncloneable advice, assuming the feasibility of copy-protecting any one sequence of maps satisfying fairly mild assumptions. In Section 5.5, we discuss one possible instantiation of the construction presented in Section 5.4.

5.1 Quantum Copy-Protection

As discussed in Section 1, quantum copy-protection is broadly the task of encoding a given function f as a quantum state ρ_f such that the following two conditions are satisfied:

- Correctness: A party with access to the quantum state ρ_f can correctly evaluate f on any input by interacting with this state in a prescribed manner.
- Security: A party with access to the quantum state ρ_f cannot create and share a bipartite quantum state with two other separated parties such that both of these parties could correctly compute f using any means available to them.

While correctness is required to hold for all inputs, security is defined with respect to a sequence of random variables $D = (D_n)_{n\in\mathbb{N}}$ jointly distributed on all functions considered by the copy-protection scheme and pairs of inputs to these functions. Security is attained if no efficient attacker can split the program state for a function f, allowing both subsequent parties to evaluate $f(x_B)$ and $f(x_C)$ with more than a negligible advantage when (f, x_B, x_C) is sampled from D.

We formalize the syntax and correctness guarantee of a copy-protection scheme in Definition 18 below. Security will then be formalized in Definition 23. Up to technical details of presentation, our definition of the syntax and correctness of a copy-protection scheme coincides with the definition given in [CLLZ21].[13]

Definition 18. Let

$$f = \left(f_n : \{0,1\}^{\kappa(n)} \times \{0,1\}^{d(n)} \to \{0,1\}^{c(n)}\right)_{n\in\mathbb{N}} \tag{30}$$

be a sequence of maps whose domains and codomains are parameterized by maps $\kappa, d, c : \mathbb{N} \to \mathbb{N}$.

A *copy-protection scheme* for f with program length $q : \mathbb{N} \to \mathbb{N}$ is a pair (G, E) of efficient (κ, q)- and $(q + d, c)$-circuit families, respectively.

A copy-protection scheme is *correct for* f if there exists a negligible function η such that

$$\langle f_n(k, x)| \, E_n(G_n(k) \otimes x) \, |f_n(k, x)\rangle \geq 1 - \eta(n) \tag{31}$$

for all $n \in \mathbb{N}$, all $k \in \{0,1\}^{\kappa(n)}$, and all $x \in \{0,1\}^{c(n)}$.

Note that the existence of a copy-protection scheme for a sequence of maps f as defined above implies that the maps κ, d, and c are polynomially bounded and efficiently computable. If this was not the case, there would not exist efficient (κ, q)- and $(q + d, c)$-circuit families.

Remark 19. The above definition can also be easily adapted to sequences of maps $(f_n)_{n\in\mathbb{N}}$ where, for all $n \in \mathbb{N}$, the domain of f_n is a subset $S_n \subseteq \{0,1\}^{\kappa(n)} \times \{0,1\}^{d(n)}$. To do so, it suffices to only require the correctness condition, Equation (31), to hold for pairs $(k, x) \in S_n$.

[13]The definitions given in [CLLZ21] states that the map f should be pseudorandom function. However, they are applicable to any such family of maps, not only pseudorandom functions.

Remark 20. It is possible for a copy-protection scheme to be correct for two different sequences of maps. Indeed, assume $f = (f_n)_{n\in\mathbb{N}}$ and $f' = (f'_n)_{n\in\mathbb{N}}$ are sequences of maps such that $f_n \neq f'_n$ for a non-zero but finite number of values of n. Then, it is straightforward to show that a copy-protection scheme (G, E) is correct for f if and only if it is correct for f'. In short, this is due to the fact that correctness is an asymptotic property and that $f_n = f'_n$ for all sufficiently large values of n.

However, this is the maximal possible discrepancy between f and f' in the sense that if (G, E) is correct for f and f', then there must exist a $\tilde{n} \in \mathbb{N}$ such that $n \geq \tilde{n} \implies f_n = f'_n$. Indeed, by the assumed correctness of the scheme for f and f', the triangle inequality for the trace distance, the Fuchs-van de Graaf inequalities, and basic properties of negligible functions, there exists a negligible function η such that

$$\begin{aligned}\frac{1}{2}\left\|f_n(k,x) - f'_n(k,x)\right\|_1 &\leq \sqrt{1 - \langle f_n(k,x)|\, \rho_{n,k,x}\, |f_n(k,x)\rangle} + \sqrt{1 - \langle f'_n(k,x)|\, \rho_{n,k,x}\, |f'_n(k,x)\rangle} \\ &\leq \eta(n)\end{aligned} \tag{32}$$

where $\rho_{n,k,x} = E_n(G_n(k) \otimes x)$ for all $n \in \mathbb{N}$, all $k \in \{0,1\}^{\kappa(n)}$, and all $x \in \{0,1\}^{d(n)}$. It then follows that $f_n = f'_n$ for all sufficiently large values of n since $\eta(n) < 1 \implies f_n(k,x) = f'_n(k,x)$.

An attack against a copy-protection scheme is a triplet of efficient circuit families (A, B, C). We will sometimes refer to the first, A, as the *splitting adversary* and the later two, B and C, as the *guessing adversaries.* This reflects their respective tasks in the game, described below, used to define and quantify the notion of security for a copy-protection scheme.

The Copy-Protection Game Let $f = (f_n)_{n\in\mathbb{N}}$ be a sequence of maps as in Definition 18, let (G, E) be a copy-protection scheme for f, and let $D = (D_n)_{n\in\mathbb{N}}$ be a sequence of random variables where, for all $n \in \mathbb{N}$, each D_n is distributed on the set $\{0,1\}^{\kappa(n)} \times \{0,1\}^{d(n)} \times \{0,1\}^{d(n)}$. The copy-protection security game, for a given $n \in \mathbb{N}$ in addition to the parameters described above, is played by a Referee against collaborating Alice, Bob, and Charlie — collectively known as the adversaries — as follows:

1. The Referee samples a triplet $(k, x_B, x_C) \leftarrow D_n$.
2. The Referee prepares the state $\rho = E_n(k)$ and gives it to Alice.
3. Alice prepares a bipartite quantum state ρ and gives one part to Bob and the other to Charlie. This is the only communication between Alice, Bob, and Charlie that occurs during the game.
4. The Referee gives x_B to Bob and x_C to Charlie. Note that Bob does not receive x_C and Charlie does not receive x_B.
5. Bob, with what they have received from the Referee and Alice, outputs a string y_B. Similarly, Charlie outputs a string y_C.
6. Alice, Bob, and Charlie win if and only if $f_n(k, x_B) = y_B$ *and* $f_n(k, x_C) = y_C$.

Formally, we will model Alice, Bob, and Charlie as efficient circuit families $A = (A_n)_{n\in\mathbb{N}}$, $B = (B_n)_{n\in\mathbb{N}}$, and $C = (C_n)_{n\in\mathbb{N}}$ respectively. Note that, at this point, we do not impose any other type of computational restraints on the Referee. In particular, we do not require the random variables $D = (D_n)_{n\in\mathbb{N}}$ to admit an efficient sampling procedure. We formalize this game and the winning probability of the adversaries in the next definition.

Definition 21. Let (G, E) be a copy-protection scheme with program lengths $q : \mathbb{N} \to \mathbb{N}$ for a sequence of maps $f = (f_n)_{n\in\mathbb{N}}$ as given in Definition 18. An attack against this scheme is a triplet (A, B, C) of efficient $(q, q_B + q_C)$-, $(d + q_B, c)$-, and $(q_C + d, c)$- circuit families, respectively, for two maps $q_B, q_C : \mathbb{N} \to \mathbb{N}$. For any $n \in \mathbb{N}$ and triplet $(k, x_B, x_C) \in \{0,1\}^{\kappa(n)} \times \{0,1\}^{d(n)} \times \{0,1\}^{d(n)}$, we define the state

$$\rho^{(A,B,C)}_{(G,E),n}(k, x_B, x_C) = (B_n \otimes C_n)\,(x_B \otimes (A_n \circ G_n)\,(k) \otimes x_C)\,. \tag{33}$$

Let $D = (D_n)_{n\in\mathbb{N}}$ be a sequence of random variables where, for each $n \in \mathbb{N}$, D_n is distributed over the set $\{0,1\}^{\kappa(n)} \times \{0,1\}^{d(n)} \times \{0,1\}^{d(n)}$. For any attack (A, B, C) against (G, E), we define its winning probability with respect to f and D, denoted $w^{(A,B,C)}_{(G,E),f,D} : \mathbb{N} \to \mathbb{R}$, as the function

$$n \mapsto \sum_{(k,x_B,x_C)\in S'_n} \Pr\left[D_n = (k, x_B, x_C)\right]\cdot\langle f_n(k, x_B), f_n(k, x_C)|\, \rho^{(A,B,C)}_{(G,E),n}(k, x_B, x_C)\,|f_n(k, x_B), f_n(k, x_C)\rangle \tag{34}$$

where $S'_n \subseteq \{0,1\}^{\kappa(n)} \times \{0,1\}^{d(n)} \times \{0,1\}^{d(n)}$ is precisely the set of all triplets (k, x_B, x_C) where both $f_n(k, x_B)$ and $f_n(k, x_C)$ are defined.

If every map f_n in f is defined on the entire set $\{0,1\}^{\kappa(n)} \times \{0,1\}^{d(n)}$, then the summation over S'_n and the $\Pr\left[D_n = (k, x_B, x_C)\right]$ factor in Equation (34) is simply an alternative expression for the expectation over sampling (k, x_B, x_C) from D_n. Such an expectation is typically how the winning probability of the adversaries is presented in the literature. However, our definition is also applicable to cases where some maps f_n are defined only on a strict subset of $\{0,1\}^{\kappa(n)} \times \{0,1\}^{d(n)}$ and where the support of D_n is a strict superset of S'_n. In these cases, the expectation would not be well defined.

It may seem unnatural to consider $w^{(A,B,C)}_{(G,E),f,D}$ for sequences f and D where D_n occasionally yields pairs (k, x) outside of the domain of f_n, but our generalization allows us to easily make sense of the winning probability of the adversaries when $f_n : \varnothing \to \{0,1\}^{c(n)}$ is the empty map. Note that it is impossible to even define a random variable distributed over the domain of f_n or, more precisely, over $S'_n = \varnothing$, in this case. Note that we can interpret appearances of the empty map in a sequence $(f_n)_{n\in\mathbb{N}}$ as corresponding to cases where the map is undefined.[14] This will be useful when we later relate the notions of copy-protection and uncloneable advice.

In general, we will only be interested in sequences f and D where the support of D_n is a subset of S'_n, unless f_n is the empty map.

Remark 22. Let $n \in \mathbb{N}$ be such that $f_n : \varnothing \to \{0,1\}^{c(n)}$ is the empty map, i.e.: $S'_n = \varnothing$. Then, adopting the convention that a sum over an empty set is 0, we have that $\omega^{(A,B,C)}_{(G,E),f,D}(n) = 0$.

The fact that $\omega^{(A,B,C)}_{(G,E),f,D}(n) = 0$ whenever f_n is the empty map, meaning that it is "in practice undefined", implies that an adversary (A, B, C) breaking the potential security of a copy-protection scheme must do so for values of n where f_n is "in practice defined". We see this as a desirable property of our definition.

[14]While it may be tempting to simply remove all instances of empty maps in a sequence (f_n) and then re-index it, yielding a new sequence (g_n), this can can have a material implications to the study of copy-protection for these maps as it impacts the scaling of the computational power permitted to schemes and adversaries. Indeed, suppose that the map f_n is defined if and only if $n = 2^k$ for some $k \in \mathbb{N}$. Under the proposed re-indexing scheme, $g_n = f_{2^n}$ for all $n \in \mathbb{N}$. In particular, if we subject the circuit families of an efficient copy-protection scheme (G, E) for $(f_n)_{n\in\mathbb{N}}$ to the same re-indexing, which would be the naive way to try to obtain a scheme for $(g_n = f_{2^n})_{n\in\mathbb{N}}$, they may become exponential-time circuit families.

Following a standard cryptographic paradigm, we now wish to define the security for a copy-protection scheme as the property that all efficient adversaries have at most a negligible advantage over some trivially attainable success probability. The issue here is to identify what precisely is the trivial success probability in the above described game.

There are a few attacks against copy-protection schemes which cannot reasonably be prevented. One such attacks, for all $n \in \mathbb{N}$, consists of having the splitting adversary A_n give the complete and unmodified program state $G_n(k)$ to the guessing party B_n and giving nothing to C_n. When the adversary B_n receives their challenge x_B, they can then honestly evaluate the program state $G_n(k)$ on x_B to obtain, with overwhelming probability assuming that the scheme (G, E) is correct for f, the proper answer. The C_n adversary, for their part, will simply output a uniformly random string sampled from $\{0,1\}^{c(n)}$, the codomain of the maps under consideration. It is easy to see that the winning probability of this trivial attack is $n \mapsto 2^{-c(n)} - \eta(n)$ for some negligible function η accounting for the possibility of an incorrect evaluation by B. Thus, we take $2^{-c(n)}$ as our trivial success probability.

Definition 23. Let $f = (f_n)_{n\in\mathbb{N}}$ be a sequence of maps as given in Definition 18, (G, E) be a copy-protection scheme for f, and $D = (D_n)_{n\in\mathbb{N}}$ be a sequence of random variables as given in Definition 21. The copy-protection scheme (G, E) is secure with respect to f and D if for every attack (A, B, C) against it, the function

$$n \mapsto w^{(A,B,C)}_{(G,E),f,D}(n) - 2^{-c(n)} \tag{35}$$

is negligible.

Note that other works present definitions based on alternate notions of trivial success probabilities. For example, [CMP20] permits the splitting adversary A_n to optimally choose which of B_n or C_n should received the unmodified program state and further allows the non-receiving guessing adversary to learn their challenge value x before making their guess for $f_n(k, x)$. As this can only increase the value of the trivial winning probability above $2^{-c(n)}$, the security notion we present here is stronger. As another example, [CLLZ21] defines security with respect to a trivial winning probability which is 0 or negligible. While this is in agreement with our definition if $n \mapsto 2^{-c(n)}$ is negligible, it is an impractical definition to use when this is not the case, such as if c is constant function.

We leave to future work a more detailed comparison of existing security notions together with their notions of trivial success probabilities.

5.2 Defining Uncloneable Advice

In this section, we briefly review the necessary preliminaries on complexity theory and then define the complexity classes **neglQP** and **neglQP**/upoly. We generally follow the definitions and conventions given in Watrous' survey of quantum complexity theory [Wat09].

First, we recall the definitions of a promise problem and of a language.

Definition 24. A *promise problem* $P = (P_0, P_1)$ is a pair of disjoint subsets $P_0, P_1 \subseteq \{0,1\}^*$. The elements of P_1 are called the *yes instances* and those of P_0 are the *no instances*. Both *yes* and *no* instances are called *problem instances*. If $P_0 \cup P_1 = \{0,1\}^*$, we say that P is a *language* and the elements of P_1 are the *words* of this language. We also establish some additional notation by overloading the symbol P twice: first as a set, then as a map.

First, we let $P = P_0 \cup P_1$ be the set of all problem instances. For all $n \in \mathbb{N}$, we also define the sets $P^n = P \cap \{0,1\}^n$, $P^n_0 = P_0 \cap \{0,1\}^n$, and $P^n_1 = P_1 \cap \{0,1\}^n$ to be all problem instances of length n, all *no* instances of length n, and all *yes* instances of length n.

Second, we let $P : P_0 \cup P_1 \to \{0,1\}$ be the unique map satisfying $P(x) = 1 \iff x \in P_1$. We identify computing the map P with the ability to *solve* the problem P. For all $n \in \mathbb{N}$, we also define $P^n : P_0^n \cup P_1^n \to \{0,1\}$ to be the unique map satisfying $P^n(x) = 1 \iff x \in P_1^n$.

Note that, with the notation of Definition 24, a problem $P = (P_0, P_1)$ is completely characterized by the resulting sequence of maps $(P^n : P_0^n \cup P_1^n \to \{0,1\})_{n\in\mathbb{N}}$.

Before proceeding to defining our novel complexity classes, we recall the classes of problems which can be solved in quantum polynomial-time with bounded errors, with or without quantum advice. As usual, we denote these classes by **BQP** and **BQP**/qpoly.

Definition 25. Let $a, b : \mathbb{N} \to \mathbb{R}$ be two maps. A promise problem P is in the class $\mathbf{BQP}(a, b)$ if there exists a polynomial-time $(n, 1)$-circuit family $(C_n)_{n\in\mathbb{N}}$ such that the following hold:

1. For all $x \in P_1$, we have that $\langle 1| C_{|x|}(x) |1\rangle \geq a(|x|)$.
2. For all $x \in P_0$, we have that $\langle 1| C_{|x|}(x) |1\rangle \leq b(|x|)$.

Definition 26. Let $a, b : \mathbb{N} \to \mathbb{R}$ be a map. A promise problem P is in the class $\mathbf{BQP}(a, b)$/qpoly if there exists a sequence of states $(\rho_n)_{n\in\mathbb{N}}$, each on $q(n)$ qubits for a polynomially bounded $q : \mathbb{N} \to \mathbb{N}$, and a polynomial-time $(q + n, 1)$-circuit family $(C_n)_{n\in\mathbb{N}}$ such that the following hold:

1. For all $x \in P_1$, we have that $\langle 1| C_{|x|}(\rho_{|x|} \otimes x) |1\rangle \geq a(|x|)$.
2. For all $x \in P_0$, we have that $\langle 1| C_{|x|}(\rho_{|x|} \otimes x) |1\rangle \leq b(|x|)$.

By a standard error-reduction-by-repetition argument, we can show that there is a large flexibility in the choices of a and b without changing the underlying class, so long as these are sufficiently bounded. Thus, we define $\mathbf{BQP} = \mathbf{BQP}(2/3, 1/3)$ and $\mathbf{BQP}/\text{qpoly} = \mathbf{BQP}(2/3, 1/3)/\text{qpoly}$.

We are now ready to define our novel complexity classes. We begin by defining the class of problems which can be solved by polynomial-time quantum computations with negligible errors. We denote this class **neglQP** and immediately note that it is equivalent to **BQP** due to standard error reduction techniques for this latter class.

Definition 27. A problem P is in the complexity class **neglQP** if there exists a polynomial-time circuit family $(C_n)_{n\in\mathbb{N}}$ and a negligible function η such that

$$x \in P \implies \langle P(x)| C_{|x|}(x) |P(x)\rangle \geq 1 - \eta(|x|). \tag{36}$$

Lemma 28. neglQP = BQP.

Proof. The inclusion $\mathbf{BQP} \subseteq \mathbf{neglQP}$ follows directly by the standard error reduction technique for **BQP**. It then suffices to show that the inclusion $\mathbf{neglQP} \subseteq \mathbf{BQP}$ also holds.

Consider a problem $P \in \mathbf{neglPQ}$ which is solved by a polynomial-time circuit family $(C_n)_{n\in\mathbb{N}}$ with negligible error η. In particular, there exists an $n_0 \in \mathbb{N}$ such that $n \geq n_0 \implies \eta(n) \leq \frac{1}{3}$. Thus, we can consider a new circuit family $(C'_n)_n$ where each C'_n for $n < n_0$ simply "hard codes" each solution to each problem instance. Otherwise, if $n \geq n_0$, we simply take $C'_n = C_n$. It is easy to see that $(C'_n)_{n\in\mathbb{N}}$ is a polynomial-time circuit family which solves P with error at most $\frac{1}{3}$. Hence, we have that $P \in \mathbf{BQP}$ and so that $\mathbf{neglQP} \subseteq \mathbf{BQP}$. □

We can now state our definition for the complexity class of problems which can be solved with negligible error by polynomial-time quantum computation with the help of *uncloneable* advice, which we denote **neglQP**/upoly.

As we have previously discussed in Section 1.1.1, a problem P is in **neglQP**/upoly if there exists a sequence of advice states $(\rho_n)_{n\in\mathbb{N}}$ satisfying the following two criteria. First, an honest user must be able to solve problem instances in P with negligible errors when given a single copy of the advice state. Second, a malicious user given a single copy of the advice state cannot share this state between two other non-communicating malicious users such that they could both solve problem instances in P with more than a negligible advantage. We highlight once again that this is analogous to the security criteria of a copy-protection scheme as set out in Definition 23.

The following definition formalizes the above.

Definition 29. A problem $P = (P_0, P_1)$ is in the complexity class **neglQP**/upoly if there exists a sequence of quantum states $(\rho_n)_{n\in\mathbb{N}}$, each on $q(n)$ qubits respectively for a polynomially bounded map $q : \mathbb{N} \to \mathbb{N}$, such that the following two conditions hold:

1. *Correctness.* There exists a polynomial-time $(q+n, 1)$-circuit family $(C_n)_{n\in\mathbb{N}}$ and a negligible function η such that
$$x \in P \implies \langle P(x)|\, C_{|x|}\left(\rho_{|x|} \otimes x\right) |P(x)\rangle \geq 1 - \eta(|x|). \tag{37}$$

2. *Uncloneability.* For each $n \in \mathbb{N}$, let
$$k_n = \begin{cases} 1 & \text{if either } P_0^n \text{ or } P_1^n \text{ is empty} \\ \frac{1}{2} & \text{else.} \end{cases} \tag{38}$$
and let D_n be a random variable distributed on $\{0,1\}^n$ such that for both $b \in \{0,1\}$ we have that
$$x \in P_b^n \implies \Pr[D_n = x] = k_n \cdot \frac{1}{\left|P_b^n\right|}. \tag{39}$$
Then, for all triplets (A, B, C) of polynomial-time $(q, q_B + q_C)$-, $(n + q_B, 1)$-, and $(q_C + n, 1)$-circuit families, respectively, there exists a negligible function η' such that for all $n \in \mathbb{N}$ satisfying $P^n \neq \varnothing$ we have that
$$\mathop{\mathbb{E}}_{(x_B,x_C)\leftarrow D_n\times D_n} \langle P(x_B), P(x_C)|\, (B_n \otimes C_n)\left(x_b \otimes A_n(\rho_n) \otimes x_c\right) |P(x_B), P(x_C)\rangle \leq \frac{1}{2} + \eta'(n). \tag{40}$$

Note that $\frac{1}{2}$ in the equation above plays the same role as the $2^{-c(n)}$ term in the definition of security for copy-protection: it captures the maximal winning probability of strategies which give the advice unmodified to one guessing party and has the other guessing party output an element uniformly at random from the appropriate codomain.

We give a few other remarks on this definition.

Remark 30. It would be possible to strengthen Definition 29 by weakening the computational constraints on the adversaries considered in uncloneability criterion. While we do not formally define these variations here, we note that the advice we construct in the proof of Theorem 35 for the particular promise problem defined in that theorem would fulfill the uncloneability criterion against any uniform adversaries, not only polynomial-time adversaries.

Remark 31. For a given problem P, the sequence of distributions $(D_n)_{n\in\mathbb{N}}$ considered in point 2 of Definition 29 is not uniquely defined. Indeed, for a particular $n \in \mathbb{N}$, the random variable D_n is uniquely defined if $P \cap \{0,1\}^n \neq \varnothing$ and is unconstrained otherwise. However, our definition is insensitive to D_n for the values of n where $P \cap \{0,1\}^n = \varnothing$ and so this ambiguity is not an issue.

Remark 32. Our definition of the sequence of distributions D in point 2 of Definition 29 could be changed, leading to possibly distinct complexity classes.

Note that while we define random variables D_n over P^n, in Equation (40) we actually use the random variables $D_n \times D_n$ over $\{0,1\}^n \times \{0,1\}^n$. In particular, any other definition of a sequence of distributions $(D'_n)_{n\in\mathbb{N}}$ on $\{0,1\}^n \times \{0,1\}^n$, provided that it satisfy

$$\forall n \in \mathbb{N} \qquad P^n \neq \varnothing \implies \Pr\big[D'_n \in P^n \times P^n\big] = 1, \tag{41}$$

could replace the sequence $(D_n \times D_n)_{n\in\mathbb{N}}$ in Definition 29 and the definition would remain coherent. We note, however, that the choice of k_n on the right-hand side of Equation (40) may no longer be the most appropriate for certain choices of distributions. Requiring the distributions to satisfy Equation (41) simply enforces the condition that the adversaries should be challenged on elements of P with certainty.

We emphasize here that the distributions D_n depend on the problem P. If we wish to consider promise problems which are not languages, then it seems natural that the distributions depend on P. Indeed, if the distributions did not depend on P, it is unclear what we should ask of the adversaries when they are challenged on inputs which are not in P. While there are sensible answers to this question, such as considering any output to be valid or not consider these cases when computing the success probability of the adversaries, we leave further discussions along these lines to future work.

We highlight three other natural ways in which we could have defined the distributions from which the problem instances (x_0, x_1) are sampled. These emerge from the four possible ways to answer the following two questions:

- Should x_0 and x_1 be equal, or sampled independently?
- Should the challenges be sampled uniformly at random from P^n, or in such a way which makes 0 and 1 equally likely to be the correct answers (when possible)?

Our choice for the distributions results from taking the latter answer to both questions.

Finally, we note that this discussion on which distributions to use when challenging the adversaries for the uncloneability criterion echoes extremely similar discussions pertaining to secure software leasing [ALP21, BJL$^+$21] and copy-protected functions [CMP20].

With Definition 29 in hand, we can now formalize the relation between copy-protection and uncloneable advice. Let (G, E) be a pair of circuit families forming a copy-protection scheme as defined in Definition 18, except that G need not be efficient, or even uniform: G can be an arbitrary circuit family. Call such a scheme a *copy-protection scheme with unconstrained generation.* Correctness and security for copy-protection schemes with unconstrained generation is exactly as defined for usual copy-protection schemes. The next theorem states that a problem P is in **neglQP**/upoly if and only if there exists a copy-protection with unconstrained generation (G, E) which is correct and secure for the sequence of maps $(P^n : P_0^n \cup P_1^n \to \{0,1\})_{n\in\mathbb{N}}$ with respect to random variables $D = (D_n)_{n\in\mathbb{N}}$ as described in Definition 29.

Theorem 33. Let P be a problem and let $D = (D_n)_{n\in\mathbb{N}}$ be a sequence of random variables as described in Definition 29. For all $n \in \mathbb{N}$, let $\tilde{P}^n : \{0,1\}^0 \times (P_0^n \cup P_1^n) \to \{0,1\}$ be the map defined by $\tilde{P}^n(\varepsilon, x) = P^n(x)$ for all $x \in P_0^n \cup P_1^n$ and let $\tilde{D}_n$ be the random variable defined by

$$\Pr\left[\tilde{D}_n = (\varepsilon, x_B, x_C)\right] = \Pr\left[D_n \times D_n = (x_B, x_C)\right]. \tag{42}$$

Then, P is in **neglQP**/upoly if and only if there exists a copy-protection scheme with unconstrained generation (G, E) for the sequence of maps $\tilde{P} = (\tilde{P}^n)_{n\in\mathbb{N}}$ which is correct and secure with respect to the distributions $\tilde{D} = (\tilde{D}_n)_{n\in\mathbb{N}}$.

Prood sketch. Assume that P is in **nelgQP**/upoly by using the sequence of advice states $(\rho_n)_{n\in\mathbb{N}}$. By the correctness condition of **neglQP**/upoly, there exists an efficient circuit family $C = (C_n)_{n\in\mathbb{N}}$ and a negligible function η such that $x \in P \implies \langle P(x)| C_n(\rho_{|x|} \otimes x) |P(x)\rangle \geq 1 - \eta(|x|)$. Moreover, there exists a circuit family $G = (G_n)_{n\in\mathbb{N}}$ such that $\frac{1}{2}\|\rho_n - G_n(\varepsilon)\|_1 \leq 2^{-n}$.[15] Consider now (G, C) as a copy-protection scheme with unconstrained generation for $(\tilde{P}^n)_{n\in\mathbb{N}}$. We note two things:

1. The correctness of the advice directly implies the correctness of (G, C) for $(\tilde{P}^n)_{n\in\mathbb{N}}$.
2. The uncloneability of the advice implies that the scheme (G, C) is secure for $(\tilde{P}^n)_{n\in\mathbb{N}}$ with respect to $(\tilde{D}_n)_{n\in\mathbb{N}}$.

   Indeed, adversaries against the scheme (G, C) are precisely adversaries against the advice states $(\rho_n)_{n\in\mathbb{N}}$ for the uncloneability criterion of **neglQP**/upoly and the winning probability of such an adversary is identical in both scenarios for those values of n satisfying $P^n \neq \varnothing$. As the advice is uncloneable, we have that for all adversaries (A, B, C) there exists a negligible function η such that $P^n \neq 0 \implies \omega^{(A,B,C)}_{(G,C),\tilde{P},\tilde{D}}(n) \leq \frac{1}{2} + \eta(n)$. Moreover, by definition, we have that $\omega^{(A,B,C)}_{(G,C),\tilde{P},\tilde{D}}(n) = 0$ whenever $P^n = \varnothing$. It follows that $n \mapsto \omega^{(A,B,C)}_{(G,C),\tilde{P},\tilde{D}} - \frac{1}{2}$ is negligible and so (G, C) is secure for $\tilde{P}$ with respect to $\tilde{D}$.

For the other direction, assume that (G, E) is a copy-protection scheme with unconstrained generation for $(\tilde{P}^n)_{n\in\mathbb{N}}$ which is correct and secure with respect to $(\tilde{D}_n)_{n\in\mathbb{N}}$. Now, for all $n \in \mathbb{N}$ define the states $\rho_n = G_n(\varepsilon)$. It follows that we can show that P is in **neglQP**/upoly with the help of the advice states $(\rho_n)_{n\in\mathbb{N}}$ since correctness and security of the copy-protection scheme directly imply correctness and security of the uncloneable advice. □

As a final remark for this section, we note that the intersection of **BQP** and **neglQP**/upoly can only contain essentially trivial problems.

Proposition 34. A problem P is in **BQP** $\cap$ **neglQP**/upoly if and only if $|P|$ is finite.

The proof of this proposition reduces to two ideas. First, if $P \in$ **BQP** then it can solved with the empty state $1 \in \mathcal{D}(\mathbb{C})$ as advice. Second, the empty state is perfectly cloneable. Or, more precisely, two copies of the empty state can be easily produced from any other state.

Proof. Assume that P is a problem such that $|P|$ is finite. Then, it is trivially true that $P \in$ **BQP**. This can be shown by considering a circuit family $(C_n)_{n\in\mathbb{N}}$ where each C_n has a "hard-coded" list of the elements of P_0^n which can then be used to solve all problem instances of length n by simply checking if the instance is in P_0^n or not. As $|P|$ is finite, this can be implemented efficiently. On the other hand, let $m \in \mathbb{N}$ be the maximal length of the elements in P, i.e.: $x \in P \implies |x| \leq m$. Consider the negligible function $\eta : \mathbb{N} \to \mathbb{R}$ defined by $\eta(n) = 1$ if $n \leq m$ and $\eta(n) = 0$ if $n > m$ and the sequence of empty states $(\rho_n = 1)_{n\in\mathbb{N}}$. Correctness and uncloneability as given in Definition 29 holds with respect to this sequence of states and this negligible function. Thus, $P \in$ **neglQP**/upoly.

[15]It may be that it is impossible to achieve $\rho_n = G_n(\varepsilon)$ for all n due to the underlying gate set from which G is constructed. This negligible error is not an issue as both correctness and security of uncloneable advice is defined up to negligible errors.

Assume that $P \in \mathbf{BQP}$ and $P \in \mathbf{neglQP}/\mathrm{upoly}$ where the second inclusion is obtained with respect to the sequence of advice states $(\rho_n)_{n\in\mathbb{N}}$. Let $C = (C_n)_{n\in\mathbb{N}}$ be an efficient $(n,1)$-circuit family and η be a negligible function such that $x \in P \implies \langle P(x)| C_{|x|}(x) |P(x)\rangle \geq 1 - \eta(|x|)$. Such a negligible function and circuit family exists by virtue of P being in $\mathbf{BQP}$. Consider now the triplet of efficient circuit families (A, C, C) where each circuit A_n simply discards its input. Finally, let $(D_n)_{n\in\mathbb{N}}$ be a sequence of random variables as defined in Definition 29. As the circuit family C solves P with negligible error on all inputs and P is assumed to be in $\mathbf{neglQP}/\mathrm{upoly}$, there exists another negligible function η' such that for all $n \in \mathbb{N}$ satisfying $P^n \neq \varnothing$ we have

$$1-2\eta(n) \leq \mathop{\mathbb{E}}_{(x_B,x_C)\leftarrow D_n\times D_n} \langle P(x_B), P(x_C)| \left(C_n \otimes C_n\right) \left(x_B \otimes A_n(\rho_n) \otimes x_C\right) |P(x_B), P(x_C)\rangle \leq \frac{1}{2}+\eta'(n), \tag{43}$$

implying that $\frac{1}{2} \leq \eta'(n) + 2\eta(n)$. Since η' and η are negligible, this inequality can hold for at most finitely many values of n. Thus, $P^n = \varnothing$ for all sufficiently large values of n, implying that $|P|$ is finite. □

5.3 A Promise Problem with Uncloneable Advice, Unconditionally

In this section, we formally describe the promise problem with uncloneable advice which was discussed in Section 1.1.4. The goal is to prove the following theorem.

Theorem 35. Let $(s_\lambda)_{\lambda\in\mathbb{N}}$ be an exponential-time ingenerable 2λ-sequence and parse each s_λ as a pair of strings of length λ, *i.e.:* $s_\lambda = (x_\lambda, \theta_\lambda)$ for $x_\lambda, \theta_\lambda \in \{0,1\}^\lambda$ for all $\lambda \in \mathbb{N}$. For each $b \in \{0,1\}$, let

$$P_b = \bigcup_{\lambda\in\mathbb{N}} \left\{(\theta_\lambda, y) \ : \ y \in \{0,1\}^\lambda \wedge x_\lambda \cdot y = b\right\}. \tag{44}$$

Then, the promise problem $P = (P_0, P_1)$ is in $\mathbf{neglQP}/\mathrm{upoly}$.

We prove this theorem by first collecting three lemmas, each encapsulating one step of the argument we made above.

We begin by recalling a lemma concerning the monogamy-of-entanglement game of [TFKW13].[16]

Lemma 36 ([TFKW13])**.** Let $n, a_B, a_C \in \mathbb{N}$ be three non-negative integers and let (A, B, C) be a triplet of $(n, a_B + a_C)$-, $(a_B + n, n)$-, and $(a_C + n, n)$-circuits, respectively. Then, we have that

$$\mathop{\mathbb{E}}_{\substack{x\leftarrow\{0,1\}^n \\ \theta\leftarrow\{0,1\}^n}} \langle x, x| (B \otimes C) \left(\theta \otimes A\left(|x^\theta\rangle\langle x^\theta|\right) \otimes \theta\right) |x, x\rangle \leq \left(\frac{1}{2} + \frac{1}{2\sqrt{2}}\right)^n < 0.86^n. \tag{45}$$

As for Lemma 12, this lemma holds for any triplet of channels, but we frame it in terms of exponential-time circuits to maintain consistency with the rest of this work and, in particular, this section. Now, we "derandomize" the above, analogously to how we previously obtained Theorem 13.

Lemma 37. Let $(s_\lambda)_{\lambda\in\mathbb{N}}$ be an exponential-time ingenerable 2λ-sequence and parse each s_λ as a pair of strings of length λ, *i.e.:* $s_\lambda = (x_\lambda, \theta_\lambda)$ for $x_\lambda, \theta_\lambda \in \{0,1\}^\lambda$ for all $\lambda \in \mathbb{N}$.

Then, for any triplet $((A_\lambda)_{\lambda\in\mathbb{N}}, (B_\lambda)_{\lambda\in\mathbb{N}}, (C_\lambda)_{\lambda\in\mathbb{N}})$ of exponential-time $(\lambda, b+c)$-, $(\lambda+b, \lambda)$-, and $(c+\lambda, \lambda)$-circuit families for maps $b, c : \mathbb{N} \to \mathbb{N}$, respectively, we have that

$$\lambda \mapsto \langle x_\lambda, x_\lambda| (B_\lambda \otimes C_\lambda) \left(\theta_\lambda \otimes A_\lambda\left(|x_\lambda^{\theta_\lambda}\rangle\langle x_\lambda^{\theta_\lambda}|\right) \otimes \theta_\lambda\right) |x_\lambda, x_\lambda\rangle \tag{46}$$

is negligible.

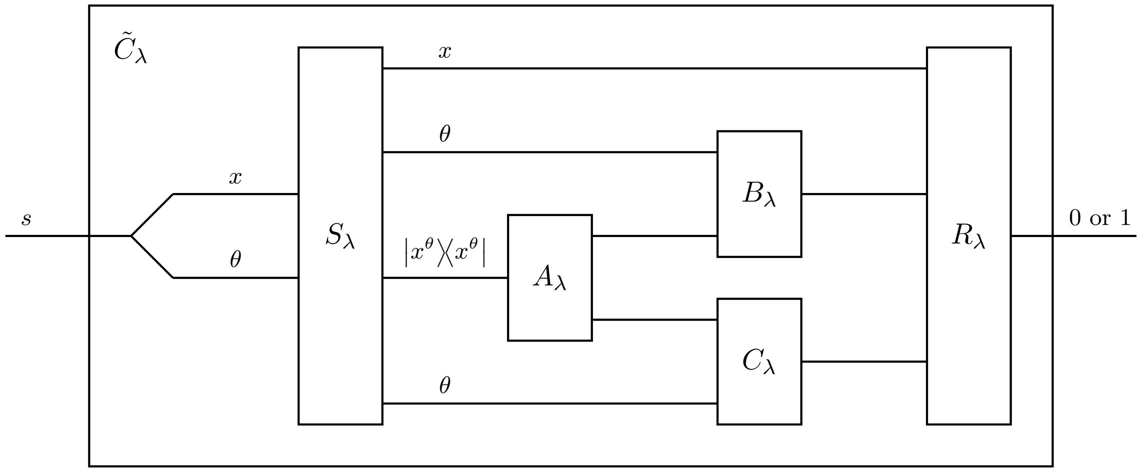


Figure 3: A schematic representation of the $\tilde{C}_\lambda$ circuit constructed in the proof of Lemma 37. The wires are labelled, when possible, with the states they are expected to carry in the context of the proof. Every wire represents λ qubits, except the initial wire, final wire, and those between the A_λ, B_λ, and C_λ circuits. These represent, respectively, 2λ, 1, b_λ, and c_λ qubits.

Proof. It suffices to apply Theorem 10 to Lemma 36. We follow the same general ideas as in the proof of Theorem 13.

Let $(S_\lambda)_{\lambda\in\mathbb{N}}$ be a polynomial-time family of $(2\lambda, 4\lambda)$-circuits such that each circuit, on input of $x\otimes\theta$, outputs the state $x\otimes\theta\otimes|x^\theta\rangle\langle x^\theta|\otimes\theta$ for any $x,\theta\in\{0,1\}^\lambda$. Let $(R_\lambda)_{\lambda\in\mathbb{N}}$ be a polynomial-time family of $(3\lambda,1)$-circuits where each circuit measures all of its input qubits in the computational bases, parses it as three strings of length λ, and outputs 1 if all three are equal and 0 otherwise.

We now consider the exponential-time family of $(2\lambda,1)$-circuits $(\tilde{C}_\lambda)_{\lambda\in\mathbb{N}}$ such that, for all $\lambda\in\mathbb{N}$, $\tilde{C}_\lambda$ is the composition of the A_λ, B_λ, C_λ, S_λ, and R_λ circuits such that the resulting channel is given by

$$\tilde{C}_\lambda = R_\lambda\circ(\mathrm{Id}_\lambda\otimes B_\lambda\otimes C_\lambda)\circ(\mathrm{Id}_\lambda\otimes\mathrm{Id}_\lambda\otimes A_\lambda\otimes\mathrm{Id}_\lambda)\circ S_\lambda. \tag{47}$$

This composition is illustrated in Figure 3. Trivial calculations shows that for all $x,\theta\in\{0,1\}^\lambda$ we have that

$$\langle 1|\,\tilde{C}_\lambda(x\otimes\theta)\,|1\rangle = \langle x,x|\,(B_\lambda\otimes C_\lambda)\left(\theta\otimes A_\lambda(\,|x^\theta\rangle\langle x^\theta|)\otimes\theta\right)|x,x\rangle \tag{48}$$

In particular, by Lemma 36,

$$\mathop{\mathbb{E}}_{\substack{x\leftarrow\{0,1\}^\lambda\\ \theta\leftarrow\{0,1\}^\lambda}}\langle 1|\,\tilde{C}_\lambda(x\otimes\theta)\,|1\rangle \le \left(\frac{1}{2}+\frac{1}{2\sqrt{2}}\right)^\lambda \tag{49}$$

is a negligible function. Thus, by Theorem 10 and the fact that $(s_\lambda)_{\lambda\in\mathbb{N}}$ is an exponential-time ingenerable 2λ-sequence, the function $\lambda\mapsto\langle 1|\,\tilde{C}_\lambda(x_\lambda\otimes\theta_\lambda)\,|1\rangle$ is also negligible which, by Equation (48), is the desired result. □

Finally, we can apply Lemma 1 to show that any exponential-time adversaries will have a negligible advantage in determining the inner product of x_λ with uniformly random strings y.

[16]This particular form of the lemma was first stated, essentially, in [BL20]. However, it is an immediate corollary of a more general result found in [TFKW13] obtained by exploiting the operational relation between measuring half an EPR pair [EPR35] and sending a random Wiesner state.

Lemma 38. Let $(s_\lambda)_{\lambda\in\mathbb{N}}$ be an exponential-time ingenerable 2λ-sequence and parse each s_λ as a pair of strings of length λ, *i.e.:* $s_\lambda = (x_\lambda, \theta_\lambda)$ for two strings $x_\lambda, \theta_\lambda \in \{0,1\}^\lambda$ for all $\lambda \in \mathbb{N}$. Then, for any triplet $((A_\lambda)_{\lambda\in\mathbb{N}}, (B_\lambda)_{\lambda\in\mathbb{N}}, (C_\lambda)_{\lambda\in\mathbb{N}})$ of exponential-time $(\lambda, b+c)$-, $(2\lambda+b, 1)$-, and $(c+2\lambda, 1)$-circuit families, respectively and for maps $b, c : \mathbb{N} \to \mathbb{N}$, there exists a negligible function η such that

$$\underset{\substack{y_B \leftarrow \{0,1\}^\lambda \\ y_C \leftarrow \{0,1\}^\lambda}}{\mathbb{E}} \langle x_\lambda \cdot y_B, x_\lambda \cdot y_C| \left(B_\lambda \otimes C_\lambda\right) \left(y_B \otimes \theta_\lambda \otimes \sigma_\lambda \otimes \theta_\lambda \otimes y_C\right) |x_\lambda \cdot y_B, x_\lambda \cdot y_C\rangle = \frac{1}{2} + \eta(\lambda) \tag{50}$$

where $\sigma_\lambda = A_\lambda\left(|x_\lambda^{\theta_\lambda}\rangle\langle x_\lambda^{\theta_\lambda}|\right)$.

Proof. In the notation of Lemma 1, we let ξ_λ be the random variable distributed on $\{0,1\}^\lambda \times \{0,1\}^\lambda$ such that $\Pr[\xi_\lambda = (x_\lambda, x_\lambda)] = 1$ and we let $\rho_{\lambda, x_\lambda, x_\lambda} = \theta_\lambda \otimes \sigma_\lambda \otimes \theta_\lambda$.

Lemma 37 implies that for every pair $((B'_\lambda)_{\lambda\in\mathbb{N}}, (C'_\lambda)_{\lambda\in\mathbb{N}})$ of exponential-time $(\lambda+b, \lambda)$- and $(c+\lambda, \lambda)$-circuits, respectively, the function

$$\lambda \mapsto \langle x_\lambda, x_\lambda| \left(B'_\lambda \otimes C'_\lambda\right)(\rho_{\lambda, x_\lambda, x_\lambda}) |x_\lambda, x_\lambda\rangle \tag{51}$$

is negligible. Since the support of ξ_λ is $\{(x_\lambda, x_\lambda)\}$, the premise of Lemma 1 is satisfied. Thus,

$$\lambda \mapsto \underset{\substack{y_B \leftarrow \{0,1\}^\lambda \\ y_C \leftarrow \{0,1\}^\lambda}}{\mathbb{E}} \langle x_\lambda \cdot y_B, x_\lambda \cdot y_C| \left(B_\lambda \otimes C_\lambda\right) \left(y_B \otimes \rho_\lambda \otimes y_C\right) |x_\lambda \cdot y_B, x_\lambda \cdot y_C\rangle - \frac{1}{2} \tag{52}$$

is also negligible. Taking η to be this map and expanding the definition of ρ_λ yields the result. □

We give a final technical lemma concerning exponential-time ingenerable sequences before proving the theorem.

Lemma 39. Let $(s_\lambda)_{\lambda\in\mathbb{N}}$ be an exponential-time ingenerable 2λ-sequence and parse each s_λ as $(x_\lambda, \theta_\lambda)$ for two strings $x_\lambda, \theta_\lambda \in \{0,1\}^\lambda$. Then, there exists a $\tilde{\lambda}$ such that $\lambda \geq \tilde{\lambda}$ implies that $x_\lambda \neq 0^\lambda$.

Proof. Assume this was not the case and $x_\lambda = 0^\lambda$ infinitely often. Consider now the polynomial-time circuit family $(C_\lambda)_{\lambda\in\mathbb{N}}$ such that C_λ samples a uniformly random $y \in \{0,1\}^\lambda$ and outputs $(0^\lambda, y)$. Then, $\langle s_\lambda| C_\lambda(\varepsilon) |s_\lambda\rangle = 2^{-\lambda}$ infinitely often. But, since $(s_\lambda)_{\lambda\in\mathbb{N}}$ is exponential-time ingenerable, there exits a polynomial p such that $\langle s_\lambda| C_\lambda(\varepsilon) |s_\lambda\rangle < p(n)2^{-2\lambda}$ for all sufficiently large values of λ. This implies that $2^\lambda < p(\lambda)$ infinitely often, a contradiction. Hence, $x_\lambda = 0^\lambda$ only finitely often. □

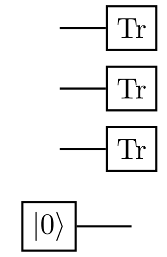
We can now give the proof of Theorem 35. Note that we will use ν as our main indexing variable and set $\nu = 2\lambda$ whenever ν is even. This will make the notation pertaining to λ a bit more consistent when invoking the previous lemma.

Proof of Theorem 35. Define the sequence of states $(\rho_\nu)_{\nu\in\mathbb{N}}$ as

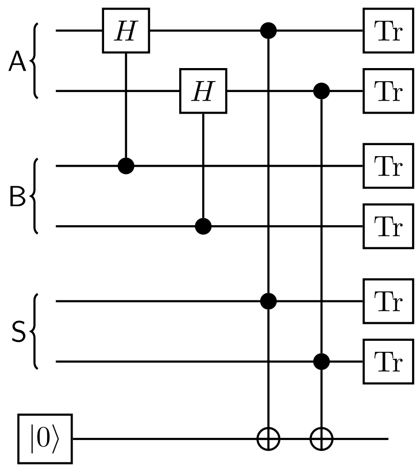
$$\rho_\nu = \begin{cases} |x_\lambda^{\theta_\lambda}\rangle\langle x_\lambda^{\theta_\lambda}| & \text{if} \quad \nu = 2\lambda \text{ for } \lambda \in \mathbb{N} \\ \varepsilon & \text{else.} \end{cases} \tag{53}$$

for all $\nu \in \mathbb{N}$. We will use these states as the advice for P. Let $a : \mathbb{N} \to \mathbb{N}$ be such that $a(\lambda)$ is the number of qubits in the advice state ρ_ν. Note that $a(\nu) = \lambda$ if $\nu = 2\lambda$ and $q(\nu) = 0$ otherwise.

Correctness. To show the correctness criterion, we give an explicit polynomial-time circuit family which solves P with certainty when given these advice states. Let $(C_\nu)_{\nu\in\mathbb{N}}$ be the polynomial-time $(a+\nu, 1)$-circuit family which is described below:



(a) The circuit C_3.



(b) The circuit C_4. The advice state is inputted in the A register and the problem instance in the other registers.

Figure 4: The circuits C_ν for $\nu = 3, 4$ used in the proof of Theorem 35 to demonstrate correctness.

- If ν is odd, C_ν discards the input and outputs a single qubit in the 0 state.
- If $\nu = 2\lambda$ is even, C_ν does the following:
  1. Parse the $q(\nu) + \nu = 3\lambda$ input qubits as three registers $\mathsf{A} \otimes \mathsf{B} \otimes \mathsf{S}$ of λ qubits each.
  2. For each $i \in [\lambda]$, apply a Hadamard gate H on the i-th qubit of A conditioned on the i-th qubit of B.
  3. Using an extra qubit initialized in the 0 state, which we will call the R register, compute the inner product of the A and S registers and store the result in the R register. We do this by applying λ Toffoli gates on R, each conditioned on the i-th qubits of A and S, respectively, for each $i \in [\lambda]$.
  4. Discard the A, B, and S registers and output the R register.

Examples of these circuits, for $\nu = 3$ and $\nu = 4$, are illustrated in Figure 4. For all $\lambda \in \mathbb{N}$ and all string $y \in \{0,1\}^\lambda$, a trivial calculation yields

$$C_{2\lambda}\left(\rho_{2\lambda} \otimes \theta_\lambda \otimes y\right) = C_{2\lambda}\left(|x_\lambda^{\theta_\lambda}\rangle\langle x_\lambda^{\theta_\lambda}| \otimes \theta_\lambda \otimes y\right) = x_\lambda \cdot y = P(\theta_\lambda, y) \tag{54}$$

Thus, for any $z \in P$, which must be of the form $z = (\theta_\lambda, y)$ for some $y \in \{0,1\}^\lambda$, we have that $C_{|z|}$ correctly computes $P(z)$ with certainty when also given the advice state $\rho_{|z|}$. Hence, the correctness criterion is satisfied.

Uncloneability. Our first step in showing that these advice states satisfy the uncloneability criterion is to gain a better understanding of the sequence of distributions $(D_\nu \times D_\nu)_{\nu \in \mathbb{N}}$, as given in Definition 29, for this promise problem.

Recall that by Lemma 39, there are only finitely many instances where $x_\lambda = 0^\lambda$. This implies that for each $b \in \{0,1\}$ and all sufficiently large values of λ, the set $P_b^{2\lambda}$ is exactly half of $P^{2\lambda}$. Indeed, this follows from the fact that $\left|\{y \in \{0,1\}^\lambda \ : \ x \cdot y = b\}\right|$ is $2^{\lambda-1}$ precisely when $x \neq 0^\lambda$. This is to say that, for all sufficiently large λ, there are always as many yes instances as no instances of length 2λ. In particular, for all sufficiently large values of λ, the random variable $D_{2\lambda}$ is uniformly random over $P^{2\lambda} = \{\theta_\lambda\} \times \{0,1\}^\lambda$.

Now, consider a triplet $((A_\nu)_{\nu\in\mathbb{N}}, (B_\nu)_{\nu\in\mathbb{N}}, (C_\nu)_{\nu\in\mathbb{N}})$ of polynomial-time $(q, b+c)$-, $(2\lambda + b, 1)$-, and $(c + 2\lambda, 1)$-circuit families, respectively and for maps $b, c : \mathbb{N} \to \mathbb{N}$. Let $v : \mathbb{N} \to \mathbb{R}$ be the function given by

$$v(\lambda) = \mathop{\mathbb{E}}_{(z_B, z_C)\leftarrow D_{2\lambda}\times D_{2\lambda}} \langle P(z_B), P(z_C)| \left(B_{2\lambda} \otimes C_{2\lambda}\right) \left(z_B \otimes A_{2\lambda}\left(\rho_{2\lambda}\right) \otimes z_C\right) |P(z_B), P(z_C)\rangle . \quad (55)$$

In other words, $v(\lambda)$ is the probability that the triplet of circuit families split and successfully using the advice state to solve problem instances in $P^{2\lambda}$ sampled according to $D_{2\lambda} \times D_{2\lambda}$. To show that the uncloneability criterion is satisfied, it suffices so show the existence of a negligible function η such that $v(\lambda) \leq \frac{1}{2} + \eta(\lambda)$ as, in the notation of Definition 29, it will then suffice to take η' to be function such that $\eta'(\nu) = 0$ if ν is odd and $\eta'(\lambda)$ if $\nu = 2\lambda$ is even. Clearly, this η' will be negligible if v is negligible.

For $L \in \{A, B, C\}$, let $L'_\lambda = L_{2\lambda}$ and note that $(L_\lambda)_{\lambda\in\mathbb{N}}$ is a polynomial-time family of circuits. Now, since the random variable $D_{2\lambda}$ is uniformly distributed on $\{(\theta_\lambda, y) : y \in \{0,1\}^\lambda\}$ for all sufficiently large values of λ, we have that

$$v(\lambda) = \mathop{\mathbb{E}}_{\substack{y_B\leftarrow\{0,1\}^\lambda \\ y_C\leftarrow\{0,1\}^\lambda}} \langle x_\lambda \cdot y_B, x_\lambda \cdot y_C| \left(B'_\lambda \otimes C'_\lambda\right) \left(\theta_\lambda \otimes y_B \otimes A'_\lambda(\rho_{2\lambda}) \otimes \theta_\lambda \otimes y_C\right) |x_\lambda \cdot y_B, x_\lambda \cdot y_C\rangle \quad (56)$$

for all sufficiently large values of λ. Up to performing the swapping map $\theta_\lambda \otimes y_B \mapsto y_B \otimes \theta_\lambda$ on the first 2λ qubits given to B'_λ, an operation which can easily be incorporated into this circuit, Lemma 38 implies the existence of a negligible function $\tilde{\eta}$ such that the right-hand side of the above equation is precisely $\frac{1}{2} + \tilde{\eta}(\lambda)$. Since $v(\lambda)$ is equal to $\frac{1}{2} + \tilde{\eta}(\lambda)$ for all sufficiently large values of λ, there exists a function $\eta : \mathbb{N} \to \mathbb{R}$ which differs from $\tilde{\eta}$ on only finitely many inputs such that $v = \frac{1}{2} + \eta$. Since η differs only finitely often from a negligible function, it is itself negligible. □

Remark 40. If the language in Theorem 35 is instantiated with an exponential-time ingenerable sequence $(s_n)_{n\in\mathbb{N}}$ which satisfies the stronger notion of being *computably* ingenerable, then the advice states given in the proof also satisfy uncloneability against uniform adversaries.

On the other hand, if the language is instantiated with an exponential time ingenerable sequences which can be computed in triple-exponential time, then we note that the advice states can be generated by quantum circuits described by a Turing machine running in triple-exponential time. By Theorem 9, such sequences exist.

Note that the advice states which we give in the proof of Theorem 35 are quantum but, in some sense, the advice is actually the *classical* strings $(x_n)_{n\in\mathbb{N}}$. We encode this classical advice into a quantum state only to achieve uncloneability, not additional computational power. With this in mind, the following proposition formalizes the idea that uncloneable advice need not be more powerful than classical advice.

Proposition 41. The intersection of **neglQP**/upoly and **P**/poly, where the latter is the class of promise problems which can be solved in polynomial-time by a classical deterministic Turing machine with access to polynomially many classical bits of advice, is non-empty.

Proof sketch. The promise problems constructed in Theorem 35, which states that these problems are in **neglQP**/upoly, can also be seen to be in **P**/poly. Indeed, if the problem is instantiated with the exponential-time ingenerable $2n$-sequence $(s_n = (\theta_n, x_n))_{n\in\mathbb{N}}$, then the classical advice strings $(s'_n)_{n\in\mathbb{N}}$ defined by

$$s'_n = \begin{cases} x_n & \text{if} \quad n \text{ is even} \\ \varepsilon & \text{else} \end{cases} \tag{57}$$

are sufficient to show that the problem constructed in Theorem 35 is also in $\mathbf{P}$/poly. □

Similarly, it is clear that if a classical party has enough computational power to generate the sequence $(s_n)_{n\in\mathbb{N}}$ or, more precisely, the sequence $(x_n)_{n\in\mathbb{N}}$, then it can solve the promise problem constructed in Theorem 35 from $(s_n)_{n\in\mathbb{N}}$.

Proposition 42. The intersection of **neglQP**/upoly and **EEEXP**, where the latter is the class of promise problems which can be solved in triple-exponential time by a classical deterministic Turing machine, is non-empty.

Proof sketch. Some instantiations of the promise problems constructed in Theorem 35, which states that these problems are in **neglQP**/upoly, can be seen to be in **EEEXP**. Indeed, if the exponential-time ingenerable sequence $(s_n)_{n\in\mathbb{N}}$ used in the construction is computable in triple-exponential time by a classical deterministic Turing machine, then the resulting promise problem is in **EEEXP**. By Theorem 9, such sequences exist. □

Propositions 41 and 42 immediately point to an open question which we believe is of interest: Are there non-trivial problems, i.e.: with infinitely many *yes* and *no* instances, in the intersection of **neglQP**/upoly and **EEXP**, the class of problems which can be solved by classical deterministic Turing machines in double exponential time? What about in the intersection of **neglQP**/upoly and **EXP** or any other classical deterministic super-polynomial time complexity classes? These questions appear to reflect those considered in Section 4 about gaps in the complexity of generating and cloning sequences of states.

5.4 A Language with Uncloneable Advice, Under Certain Assumptions

In this section, we describe a class of languages which admit uncloneable advice under the assumption that it is possible to copy-protect certain families of maps. We proceed in two steps. First, we give a general template in Definition 43 for the class of languages we consider. Each language from this template is parameterized by a sequence of maps and is illustrated in Figure 5. Second, in Theorem 45, we establish sufficient conditions on the parametrizing maps which ensure that the resulting language admits uncloneable advice. We comment, in the next section, on certain specific instantiations of this language admitting uncloneable advice based on existing constructions for the copy-protection of pseudorandom functions.

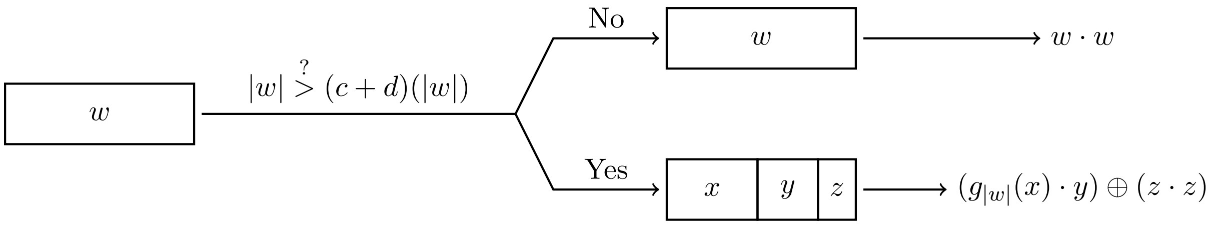
Definition 43. Let $g = (g_n : \{0,1\}^{d(n)} \to \{0,1\}^{c(n)})_{n\in\mathbb{N}}$ be a sequence of maps whose domains and codomains are parameterized by $d, c : \mathbb{N} \to \mathbb{N}$. We define the language $L^g = (L^g_0, L^g_1)$ as follows:

- If $w \in \{0,1\}^*$ is such that $|w| \leq (d+c)(|w|)$, then

$$w \in L^g_b \iff w \cdot w = b. \tag{58}$$

- If $w \in \{0,1\}^*$ is such that $|w| > (d+c)(|w|)$, we parse w as (x, y, z) where x is composed of the first $d(|w|)$ bits of w, y of the next $c(|w|)$ bits, and z of the remaining bits. Then,

$$w = (x, y, z) \in L^g_b \iff (g_{|w|}(x) \cdot y) \oplus (z \cdot z) = b. \tag{59}$$



w
$|w| \overset{?}{>} (c+d)(|w|)$
No
w
$w \cdot w$
Yes
x
y
z
$(g_{|w|}(x) \cdot y) \oplus (z \cdot z)$

Figure 5: An illustration of an algorithm which determines if a given string $w \in \{0,1\}^*$ is in the set L_0^g or the set L_1^g where these are as given in Definition 43. The single bit produced, at the right of the diagram, determines to which set w belongs. In the case where the criterion $|w| > (d+c)(|w|)$ is met, we parse w as a triplet of strings (x, y, z), each of the unique length ensuring that $(g_{|w|}(x) \cdot y) \oplus (z \cdot z)$ is well defined.

We note a nice property of L^g which is independent of the choice of g, namely that exactly half of all bit strings of a given non-zero length are in the language.

Lemma 44. Let $L^g = (L_0^g, L_1^g)$ be as given in Definition 43. Then, for all $n \geq 1$ we have that

$$|L_0^g \cap \{0,1\}^n| = |L_1^g \cap \{0,1\}^n| = 2^{n-1}. \tag{60}$$

In other words, for any given non-zero length n, there are as many *yes* instances as there are *no* instances of length n in L^g.

Proof. This follows immediately from the fact that for all $n \geq 1$ and $b \in \{0,1\}$ we have that

$$|\{s \in \{0,1\}^n \ : \ s \cdot s = b\}| = 2^{n-1}. \tag{61}$$

□

We now establish conditions under which the language L^g defined above admits uncloneable advice.

Theorem 45. Let $f = (f_n : \{0,1\}^{\kappa(n)} \times \{0,1\}^{d(n)} \to \{0,1\}^{c(n)})_{n \in \mathbb{N}}$ be a sequence of maps whose domains and codomains are parameterized by maps $\kappa, d, c : \mathbb{N} \to \mathbb{N}$ such that the following conditions are satisfied:

- There exists an $n' \in \mathbb{N}$ such that $n \geq n' \implies d(n) + c(n) < n$ and $c \in \omega(\log)$.
- There exists a copy-protection scheme (G, E) for f which is correct and secure.

Let $k = (k_n)_{n \in \mathbb{N}}$ be an exponential-time ingenerable κ-sequence and let $g = (g_n)_{n \in \mathbb{N}}$ be the sequence of maps where $g_n = f_n(k_n, \cdot)$ for all $n \in \mathbb{N}$. Then, the language L^g is in $\mathbf{neglQP}/\mathrm{upoly}$.

At a high level, the proof proceeds as follows: The advice states will be the program states generated by the copy-protection scheme for the maps in the sequence g. Correctness follows trivially from the correctness of the copy-protection scheme. Uncloneability is a bit more involved. Consider only the cases where $n \geq n'$ and so any string w will be parsed as (x, y, z). The first step is to argue that the z component does not really contribute to the difficulty of determining if w is in the language. The only difficult part is for both guessers to be able to correctly and simultaneously compute $g_n(x) \cdot y$. Since the y's are independently sampled for both guessers, the result of Kundu and Tan (Lemma 1) tells us that if both parties can only compute $g_n(x)$ with a negligible probability of success, then they can simultaneously correctly compute this inner product with at most a negligible advantage above $\frac{1}{2}$. How can we show that this probability of simultaneously guessing $g_n(x)$ on independent values of x is negligible? It suffices to derandomize the copy-protection game with the exponential-time ingenerable sequence $(k_n)_{n\in\mathbb{N}}$. More precisely, since the advice states are produced from a secure copy-protection scheme for maps where the trivial guessing probability 2^{-c} is trivial (as $c \in \omega(\log)$), we know that any efficient adversaries will fail to simultaneously and correctly compute $f_n(k', x)$ with more than a negligible probability if the key k' is uniformly random (but identical for both adversaries) and where x is uniformly random and independent for both adversaries. By Theorem 10, the probability will remain negligible even when we do not sample k' uniformly at random, but rather fix it to the n-th element of an exponential-time ingenerable κ-sequence.

Proof. Assume the copy-protection scheme (G, E) has program length q. We will consider the sequence of states

$$\rho = (\rho_n = G_n(k_n))_{n\in\mathbb{N}}\,, \tag{62}$$

which are the program states produced by the copy-protection scheme (G, E) for the maps $(g_n)_{n\in\mathbb{N}}$, as our advice for this language.

Correctness. We first show correctness. This follows directly from the assumed correctness of the copy-protection scheme (G, E). In short, we implement the algorithm shown in Figure 5 while using ρ_n and E_n to evaluate g_n. More explicitly, let $C = (C_n)_{n\in\mathbb{N}}$ be the efficient $(q + n, 1)$-circuit family where each C_n implements the following algorithm on input of $\rho \otimes w$:

1. Verify if $n \leq d(n) + c(n)$.
   - If this inequality holds, output the bit $w \cdot w$ and terminate.
   - If this inequality does not hold, execute steps 2 to 4 below.
2. Parse w as a triplet (x, y, z) where x is the first $d(n)$ bits, y the subsequence $c(n)$ bits, and z the remaining $n - d(n) + c(n)$ bits.
3. Run $E_n(\rho_n \otimes x)$ and measure the output in the computational basis. Let y' denote this result.
4. Output the bit $(y' \cdot y) \oplus (z \cdot z)$ and terminate.

If $n \leq d(n) + c(n)$, then $C_n(\rho_n \otimes w)$ outputs the correct bit, namely $w \cdot w$, with probability 1. Else, $C_n(\rho_n \otimes w)$ outputs the correct bit with a probability at least

$$\langle g_n(x)|\, E_n(\rho_n \otimes x)\, |g_n(x)\rangle = \langle f_n(k_n, x)|\, E_n\left(G_n(k_n) \otimes x\right) |f_n(k_n, x)\rangle \tag{63}$$

which is the probability that step 3 correctly evaluates to $g_n(x)$. Since (G, E) is correct, there exists a negligible function η_c such that this probability is at least $1 - \eta_c(n)$. Hence, the correctness criterion is satisfied.

Uncloneability. First, let's set up an adversary and state what is sufficient to show to demonstrate the uncloneability criterion.

Let (A, B, C) be a triplet of polynomial-time $(q, q_B + q_C)$-, $(d + q_B, 1)$-, and $(q_C + d, 1)$-circuit families, respectively, for maps $q_B, q_C : \mathbb{N} \to \mathbb{N}$. Let D_n be the random variable distributed on the set $\{0, 1\}^n$ as defined in Definition 29 for the language L^g. Note that, by Lemma 44, D_n is uniformly distributed. Let

$$v(n) = \mathop{\mathbb{E}}_{w_B, w_C \leftarrow D_n} \langle L^g(w_B), L^g(w_C)|\, (B_n \otimes C_n)\, (w_B \otimes A_n(\rho_n) \otimes w_C)\, |L^g(w_B), L^g(w_C)\rangle \tag{64}$$

be the probability that the adversaries (A, B, C) succeed in sharing the advice state ρ_n and use it to correctly determine the membership of two independent problem instances of length n sampled according to D_n. It suffices to show the existence of a negligible function η such that $v \leq \frac{1}{2} + \eta$. In fact, it suffices to show that this inequality holds for all $n \geq n'$ where, we recall, n' is a threshold after which $n < d(n) + c(n)$ always holds. Assume from now on that $n \geq n'$ is always satisfied and parse every $w \in \{0,1\}^n$ as a triplet (x, y, z) where x is the first $d(n)$ bits, y the subsequent $c(n)$ bits, and z the remaining bits.

Now, as a first step, we argue that the z component of a problem instances essentially does not matter. In the process, we will express $v(n)$ in a form which will be closer to something to which we can apply Lemma 1.

Let B' be a circuit family such that for all $n \geq n'$, the circuit B'_n implements the following algorithm on input of $\rho \otimes x \otimes y$:

- Sample a uniformly random $z \leftarrow \{0,1\}^{n-d(n)-c(n)}$.
- Run $B_n(\rho \otimes x \otimes y \otimes z)$ and measure the output in the computational basis. Let b denote the result of this measurement.
- Output the bit $b \oplus (z \cdot z)$ and terminate.

(For completeness, we can assume that if $n < n'$ then B'_n simply discards its input and outputs 0. These cases do not matter in our analysis.) Note that B' is an efficient family of circuits. Define the circuit family $C' = (C'_n)_{n\in\mathbb{N}}$ analogously with respect to the family C. Note that B'_n outputs the bit b' if and only if its execution of B_n outputs the bit $b' \oplus (z \cdot z)$ and similarly for C'_n. Hence, since D_n is uniformly distributed, we have that $v(n)$ can also be expressed as

$$\mathop{\mathbb{E}}_{\substack{x_B,x_C \leftarrow \{0,1\}^{d(n)} \\ y_C,y_C \leftarrow \{0,1\}^{c(n)}}} \langle g_n(x_B) \cdot y_B, g_n(x_C) \cdot y_C | \, (B'_n \otimes C'_n)(y_B \otimes \sigma_n^{x_B,x_C} \otimes y_C) \, | g_n(x_B) \cdot y_B, g_n(x_C) \cdot y_C \rangle \tag{65}$$

where

$$\sigma_n^{x_B,x_C} = x_B \otimes A_n(\rho_n) \otimes x_C. \tag{66}$$

To show that v is at most negligibly more than $\frac{1}{2}$, it now suffices by Lemma 1 to show that for all pairs (B'', C'') of efficient $(d + q_B, c)$- and $(q_C + d, c)$-circuit families, respectively, we have that

$$n \mapsto \mathop{\mathbb{E}}_{x_B,x_C \leftarrow \{0,1\}^{d}(n)} \langle g_n(x_B), g_n(x_C) | \, (B''_n \otimes C''_n)(\sigma_n^{x_B,x_C}) \, | g_n(x_B), g_n(x_C) \rangle \tag{67}$$

is a negligible map.

Since (G, E) is a secure copy-protection scheme and 2^{-c} is negligible, we have that

$$\mathop{\mathbb{E}}_{k \leftarrow \{0,1\}^{\kappa(n)}} \mathop{\mathbb{E}}_{x_B,x_C \leftarrow \{0,1\}^{d}(n)} \langle f_n(k, x_B), f_n(k, x_C) | \, (B''_n \otimes C''_n)(x_B \otimes A_n(G_n(k)) \otimes x_C) \, | f_n(k, x_B), f_n(k, x_C) \rangle \tag{68}$$

is negligible in n and so, by Theorem 10 and the fact that $\kappa \in \omega(\log)$,[17] we can fix the keys k to be the elements of the exponential-time ingenerable sequence $(k_n)_{n\in\mathbb{N}}$ and obtain that

$$n \mapsto \mathop{\mathbb{E}}_{x_B,x_C \leftarrow \{0,1\}^{d(n)}} \langle f_n(k_n, x_B), f_n(k_n, x_C) | \, (B''_n \otimes C''_n)(x_B \otimes A_n(G_n(k_n)) \otimes x_C) \, | f_n(k_n, x_B), f_n(k_n, x_C) \rangle \tag{69}$$

[17]If $\kappa \notin \omega(\log)$, then $2^{-\kappa}$ is non-negligible. Thus, a triplet of adversaries can break the assumed security of the copy-protection scheme (G, E) by sampling a key universally at random and using it, with the honest generation and evaluation algorithms, to correctly evaluate the function correctly with non-negligible probability. Indeed, they will have a non-negligible probability of having guessed the correct key.

is also negligible. Recalling that $f_n(k_n, \cdot) = g_n$, this is precisely what is needed.

Note that we neglect in this proof to give an explicit construction of a $(\kappa, 1)$-circuit which models the complete set-up and attack of the copy-protection scheme on input of a given key k, as is technically required to apply Theorem 10. However, this construction would simply follow the same ideas as the ones presented in the proofs of Theorem 13 and Lemma 37: The attacking circuits from B'' and C'' are placed between an initial set-up circuit, which creates and distributes the program state and challenges, and a final referee circuit, which receives the guesses and outputs 1 if and only if they are correct. This composition would yield the necessary circuits.

Recalling that $\sigma_n^{x_B,x_C} = x_B \otimes A_n(G_n(k')) \otimes x_C$ is then sufficient to complete the proof. □

5.5 Instantiating our Construction of a Language with Uncloneable Advice

Our construction of a language with uncloneable advice in the previous section did not consider an important question: Is it possible to copy-protect a sequence of functions satisfying the necessary conditions? The following theorem gives one answer to this question.

Theorem 46. If there exists a correct and secure copy-protection scheme for a pseudorandom function $f' = (f'_n)_{n\in\mathbb{N}}$ with outputs of super-logarithmic length, then there exists a sequence of maps $f = (f_n)_{n\in\mathbb{N}}$ satisfying the assumptions of Theorem 45.

Before proving this theorem, we recall the following result from [CLLZ21]. It establishes a set of sufficient conditions, which we do not formalize in this work, for the copy-protection of a certain construction of pseudorandom functions presented in that work. By the previous theorem, it follows that these are also sufficient conditions for the existence of a language with uncloneable advice.

Theorem 47 ([CLLZ21])**.** Assuming the existence of post-quantum indistinguishable obfuscation, one-way functions, and compute-and-compare obfuscation for the class of unpredictable distributions, there exists a secure and correct copy-protection scheme for a specific construction of pseudorandom functions where the length of the keys, inputs, and outputs are non-constant polynomials.

For completion, we recall the definition of a pseudorandom function. We will not be using this definition in our work. Up to specifying efficient circuit families as our computational model and restricting the domains and codomains to be sets of bit strings, this is the definition given by Zhandry [Zha12]. In short, a keyed function is pseudorandom if, when given oracle access, it cannot be distinguished from a truly random function.

Definition 48. A sequence of maps $f = (f_n : \{0,1\}^{\kappa(n)} \times \{0,1\}^{d(n)} \to \{0,1\}^{c(n)})_{n\in\mathbb{N}}$ is a pseudorandom function if for all efficient $(0,1)$-circuit families C^{O_g} having access to an oracle computing maps of the form $g : \{0,1\}^{d(n)} \to \{0,1\}^{c(n)}$ it holds that

$$\lambda \mapsto \left| \mathop{\mathbb{E}}_{g} \langle 1| C_n^{O_g}(\varepsilon) |1\rangle - \mathop{\mathbb{E}}_{k\in\{0,1\}^{\kappa(n)}} \langle 1| C_n^{O_{f(k,\cdot)}} |1\rangle \right| \tag{70}$$

is a negligible function and where we understand $\mathbb{E}_g$ to represent the expectation over the uniformly random choice of a map $g : \{0,1\}^{d(n)} \to \{0,1\}^{c(n)}$. Moreover, we require that there exists an efficient circuit family F such that $F_n(k \otimes x) = f_n(k, x)$ for all $n \in \mathbb{N}$ and that κ be efficiently computable.

Note that the PRF f' assumed in Theorem 46 do not necessarily satisfy the first condition of Theorem 45, which is to say that there is no guarantee that $d(n) + c(n)$ will eventually always be strictly smaller than n. In fact, this is never satisfied for the PRF considered in [CLLZ21]. Thus, we need to show that we can "slowdown" the growth of this value, while keeping the correctness and, more importantly, the security of the scheme. The way we will do this is by repeating certain elements of the sequence $f' = (f'_n)_{n\in\mathbb{N}}$ multiple times in succession to obtain a new sequence of maps $f = (f_n)_{n\in\mathbb{N}}$. This will be parametrized by a re-indexing map $\gamma : \mathbb{N} \to \mathbb{N}$ which will, in general, *not* be injective and where $f_n = f'_{\gamma(n)}$. In some sense, γ "slows down" the rate of growth of the security parameter for the maps. The content of the following lemma establishes conditions where this transformation preserves correctness and security.

Lemma 49. Let (G, E) be a copy-protection scheme for a sequence of maps f, as defined in Equation (30), which is correct and secure and where 2^{-c} is a negligible map. Let $\gamma : \mathbb{N} \to \mathbb{N}$ be a non-decreasing computable map in $\Omega(\lambda^{r_l})$ and $\mathcal{O}(\lambda^{r_u})$ for some $r_l, r_u \in \mathbb{R}^+$. Let $G^\gamma = (G^\gamma_\lambda = G_{\gamma(\lambda)})_{\lambda\in\mathbb{N}}$, $E^\gamma = (E^\gamma_\lambda = E_{\gamma(\lambda)})_{\lambda\in\mathbb{N}}$, and $f^\gamma = (f^\gamma_\lambda = f_{\gamma(\lambda)})_{\lambda\in\mathbb{N}}$. Then, (G^γ, E^γ) is a copy-protection scheme for f^γ which is correct and secure.

Moreover, $2^{-c\circ\gamma}$ is also negligible, as it is simply $2^{-c} \circ \gamma$ and $\gamma \in \Omega(\lambda^{r_l})$.

Proof. First, note that since γ is efficiently computable and upper-bounded by a polynomial, as it is in $\mathcal{O}(\lambda^{r_u})$, then G^γ and E^γ are efficient circuit families. The correctness of (G^γ, E^γ) follows trivially from the correctness of (G, E) and the fact that the map $\eta \circ \gamma$ is negligible if η is negligible as $\gamma \in \Omega(\lambda^{r_l})$. We move on to showing the security.

Let (A', B', C') be an attack against (G^γ, E^γ). We construct an attack (A, B, C) against the original scheme (G, E) as follows. For completion, assume we have an existing attack $(\tilde{A}, \tilde{B}, \tilde{C})$ against (G, E). The specifics of this attack do not matter for this proof; it will only be used to fill in some gaps and ensure that all of our objects are well defined. We use n to index the security parameter of (A', B', C') and λ to index the one of (A, B, C).

For all $\lambda \in \mathbb{N}$, define A_λ as follows:

- If $\gamma^{-1}(\lambda) = \varnothing$, then $A_\lambda = \tilde{A}_\lambda$.
- If $\gamma^{-1}(\lambda) \neq \varnothing$, then A_λ is a circuit which samples uniformly at random an $n' \in \gamma^{-1}(\lambda)$, executes $A'_{n'}$ and then gives both B_λ and C_λ a copy of n' in addition to their respective share of the output of $A'_{n'}$.

For all $\lambda \in \mathbb{N}$, we define B_λ as follows:

- If $\gamma^{-1}(\lambda) = \varnothing$, then $B_\lambda = \tilde{B}_\lambda$.
- If $\gamma^{-1}(\lambda) \neq \varnothing$, then B_λ is a circuit which reads the n' value received from A_λ and then executes the $B'_{n'}$ circuit on the remainder of the input received from A_λ.

The C_λ circuits are defined analogously to the B_λ circuits. Before proceeding, we should ensure that the (A, B, C) defined here is a triplet of *efficient* circuit families.

First, we show that the set $\gamma^{-1}(\lambda)$ is efficiently computable from λ, in the sense that there exists a polynomial-time Turing machine which, on input of 1^λ, outputs an encoding of $\gamma^{-1}(\lambda)$.

By our assumptions on the map γ, there exists a n_γ and $k_u, k_l \in \mathbb{R}^+$ such that $n \geq n_\gamma$ implies that $k_l n^{r_l} \leq \gamma(n) \leq k_u n^{r_u}$. Thus, for any $n \geq n_\gamma$,

$$n \in \gamma^{-1}(\lambda) \implies k_l \lambda^{r_l} \leq \gamma(n) = \lambda \leq k_u n^{r_u} \implies (\lambda/k_u)^{\frac{1}{r_u}} \leq n \leq (\lambda/k_l)^{\frac{1}{r_l}}. \tag{71}$$

This implies that there are at most $\max\{n_\gamma, (\lambda/k_l)^{\frac{1}{\lambda_\ell}}\} \in \mathcal{O}(\gamma^{\frac{1}{r_\ell}})$ values in $\gamma^{-1}(\lambda)$. Specifically, the possible elements of $\gamma^{-1}(\lambda)$ are the non-negative integers no greater than $\max\{n_\gamma, (\lambda/k_l)^{\frac{1}{r_\ell}}\}$. Thus, one efficient way to compute $\gamma^{-1}(\lambda)$ is to simply compute γ on each of these candidate integers and verify if the result is n. Since each computation can be done in polynomial-time and there are at most a polynomial number of candidate values, this is an efficient computation of $\gamma^{-1}(\lambda)$. Note that this reasoning also implies that $\left|\gamma^{-1}(\lambda)\right|$ is upper bounded by a polynomial in λ.

It then follows that the circuit families (A, B, C) described above are efficient. Indeed, checking if $\gamma^{-1}(\lambda)$ is empty or not can be done efficiently. If it is empty, A_λ, B_λ and C_λ are simply circuits which are already assumed to be from an efficient family. If $\gamma^{-1}(\lambda)$ is not empty, then A_λ, B_λ, and C_λ is simply $\left|\gamma^{-1}(\lambda)\right|$ circuits in parallel with a minimal overhead to route the inputs to the right circuit. Since $\left|\gamma^{-1}(\lambda)\right|$ is polynomially bounded, this remains efficient.

Thus, by the security of (G, E) and the fact that 2^{-c} is negligible, there exists a negligible function η such that

$$\mathop{\mathbb{E}}_{\substack{k \leftarrow \{0,1\}^{\kappa(\lambda)} \\ x_B, x_C \leftarrow \{0,1\}^{d(\lambda)}}} \langle f_\lambda(k, x_B), f_\lambda(k, x_C)| \left(B_\lambda \otimes C_\lambda\right) \left(x_B \otimes A_\lambda(G_\lambda(k)) \otimes x_C\right) |f_\lambda(k, x_B), f_\lambda(k, x_C)\rangle \leq \eta(\lambda) \tag{72}$$

Let $v(\lambda)$ denote the left-hand side of the above. By our construction, the attack (A, B, C) averages multiple instances of the attack (A', B', C') when possible. More precisely, if we let $v'(n)$ denote the success probability of the attack (A', B', C') for the security parameter n, we have that

$$\mathop{\mathbb{E}}_{n \in \gamma^{-1}(\lambda)} v'(n) = v(\lambda) \leq \eta(\lambda) \tag{73}$$

if $\gamma^{-1}(\lambda) \neq \varnothing$. In particular, this implies that $v'(n) \leq \left|\gamma^{-1}(\gamma(n))\right| \cdot \eta \circ \gamma(n)$. By our assumption on the map γ, $\gamma(n)$ is upper bounded by a polynomial in n, which by our previous remarks also implies that $\left|\gamma^{-1}(\gamma(n))\right|$ is upper bounded by a polynomial in n. Since γ is in $\Omega(n^{r_l})$ and is non-decreasing, we have that $\eta \circ \gamma$ is negligible. It follows that $v'(n)$ is negligible, which is the desired result. $\square$

While the previous lemma gives sufficient condition for when a "γ slowdown" maintains the security and correctness guarantees of copy-protected functions, it does not establish the existence of a suitable γ. This is done, implicitly, by the following lemma.

Lemma 50. Let $g : \mathbb{N} \to \mathbb{N}$ be a map such that $g \in \mathcal{O}(n^d)$ for a $d \in \mathbb{N}^+$. Then, there exists an efficiently computable non-decreasing map $f : \mathbb{N} \to \mathbb{N}$ such that $g \circ f$ is eventually strictly smaller than n and $f \in \Omega(\lambda^r) \cap \mathcal{O}(\lambda^r)$ for a $r \in \mathbb{R}^+$.

Proof. Let $k, n_g \in \mathbb{N}$ be such that $n \geq n_g \implies g(\lambda) \leq kn^d$. Now, let f be defined by

$$f(n) = \begin{cases} 0 & \text{if} \quad n < \sqrt[d]{k} n_g^{2d} \\ \left\lfloor \frac{1}{\sqrt[d]{k}} n^{\frac{1}{2d}} \right\rfloor & \text{else.} \end{cases} \tag{74}$$

Trivially, f is non-decreasing and efficiently computable. Then, for all $n > \max\{\sqrt[d]{k} n_g^{2d}, 1\}$, we have that

$$g \circ f(n) = g\left(\left\lfloor \frac{1}{\sqrt[d]{k_u}} n^{\frac{1}{2d}} \right\rfloor\right) \leq n^{\frac{1}{2}} < n. \tag{75}$$

It now remains to show that there exists an $r \in \mathbb{R}^+$ such that $f \in \Omega(\lambda^r)$. Note that for all sufficiently large values of λ we have that $f(\lambda) \geq \frac{1}{\sqrt[d]{k}} \lambda^{\frac{1}{2d}} - 1 \geq \frac{1}{2\sqrt[d]{k}} \lambda^{\frac{1}{2d}}$. Thus, $f \in \Omega(\lambda^{\frac{1}{2d}})$. We similarly find that $f \in \mathcal{O}(\lambda^{\frac{1}{2d}})$. $\square$

Pulling these lemmas together, we can prove Theorem 46.

Proof of Theorem 46. Let $f' = (f'_n : \{0,1\}^{\kappa(n)} \times \{0,1\}^{d(n)} \to \{0,1\}^{c(n)})_{n\in\mathbb{N}}$ be the copy-protected PRF. We know that $d + c \in \mathcal{O}(n^r)$ for some $r \in \mathbb{R}^+$ else it would not be possible to efficiently compute this PRF. Then, by Lemma 50, there exists a non-decreasing efficiently computable γ in both $\Omega(\lambda^{\frac{1}{2d}})$ and $\mathcal{O}(\lambda^{\frac{1}{2d}})$ such that $(c + d) \circ \gamma(n)$ is eventually strictly smaller than n.

Thus, by applying the transformation of Lemma 49 with this γ to f', we obtain a sequence of maps f satisfying the assumptions of Theorem 45. □

A Comparing Ingenerable and Martin-Löf Random Sequences

The goal of this section is to show that the concepts of ingenerable sequences and of Martin-Löf random sequences are distinct but related. For the rest of the appendix, "ingenerable" is understood to mean "computably ingenerable".

- For reasonable choices of ℓ, there exists weakly ingenerable ℓ-sequences which are not ingenerable (Theorem 51).
- For any non-constant polynomial ℓ, every Martin-Löf random sequence yields a weakly ingenerable ℓ-sequence via a natural bijection (Theorem 60).
- There exists ingenerable ℓ-sequences which do not correspond to Martin-Löf random sequences under the same natural bijection mentioned above (Theorem 59).

For any non-constant polynomial ℓ, Figure 6 shows the relation obtained between the sets of weakly ingenerable ℓ-sequences, ingenerable ℓ-sequences, and Martin-Löf random sequences obtained from the results above.

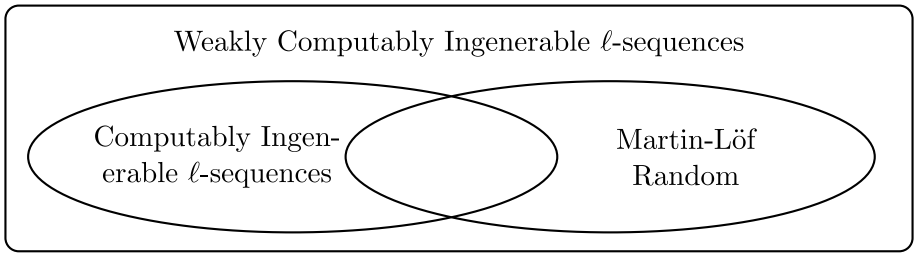

Weakly Computably Ingenerable ℓ-sequences
Computably Ingenerable ℓ-sequences
Martin-Löf Random

Figure 6: The relation between the sets of weakly computably ingenerable ℓ-sequences, ingenerable ℓ-sequences, and Martin-Löf random sequences, via the cut_ℓ bijection, for any non-constant polynomial $\ell : \mathbb{N} \to \mathbb{N}$. All three sets are distinct, but the precise relation between ingenerable ℓ-sequences and Martin-Löf random sequences remains unknown.

Before moving on to considering Martin-Löf randomness, we separate the notions of ingenerability and weak ingenerability, except in the trivial cases where all ℓ-sequences are ingenerable.

Theorem 51. Let $\ell : \mathbb{N} \to \mathbb{N}$ be a computable map not in $\mathcal{O}(\log)$. Then, there exists a weakly ingenerable ℓ-sequence which is not ingenerable.

Proof. Let $(s_n)_{n\in\mathbb{N}}$ be an ingenerable ℓ-sequence. For all $n \in \mathbb{N}$, we define

$$s_n^0 = \begin{cases} 0^{\ell(n)} & \text{if } n \text{ is even} \\ s_n & \text{if } n \text{ is odd} \end{cases} \quad \text{and} \quad s_n^1 = \begin{cases} s_n & \text{if } n \text{ is even} \\ 0^{\ell(n)} & \text{if } n \text{ is odd} \end{cases} \tag{76}$$

We first show that at least one of $(s_n^0)_{n\in\mathbb{N}}$ or $(s_n^1)_{n\in\mathbb{N}}$ is not ingenerable. Aiming for a contradiction, assume both are ingenerable. Let $(C_n)_{n\in\mathbb{N}}$ be a uniform $(0,\ell)$-circuit family such that C_n simply outputs $\left|0^{\ell(n)}\middle\rangle\middle\langle 0^{\ell(n)}\right|$ for all $n \in \mathbb{N}$. As we have assumed that both sequences are ingenerable, there exists polynomials p_b and integers $\tilde{n}_b$ such that

$$n \geq \tilde{n}_b \implies \langle s_n^b | \, C_n(\varepsilon) \, | s_n^b \rangle < p_b(n) \cdot 2^{-\ell(n)}. \tag{77}$$

for both values of $b \in \{0,1\}$. Taking $\tilde{n} = \tilde{n}_0 + \tilde{n}_1$ and $p = p_0 + p_1$, this implies that

$$n \geq \tilde{n} \implies \langle s_n^b | \, C_n(\varepsilon) \, | s_n^b \rangle < p(n) \cdot 2^{-\ell(n)} \tag{78}$$

for both $b \in \{0,1\}$. Since at least one of s_n^0 or s_n^1 is always the all-zero string, which precisely what is produced by C_n, this implies that

$$n \geq \tilde{n} \implies 1 < p(n) \cdot 2^{-\ell(n)} \tag{79}$$

from which we can conclude that $\ell \in \mathcal{O}(\log)$, a contradiction. Hence, at least one of the two sequences is not ingenerable.

Next, we show that $(s_n^b)_{n\in\mathbb{N}}$ is weakly ingenerable for both values of $b \in \{0,1\}$. Since $(s_n)_{n\in\mathbb{N}}$ is ingenerable, there exist a polynomial p such that for any uniform $(0,\ell)$-circuit family $(C_n)_{n\in\mathbb{N}}$ there is a $\tilde{n} \in \mathbb{N}$ such that

$$n \geq \tilde{n} \implies \langle s_n | \, C_n(\varepsilon) \, | s_n \rangle < p(n) \cdot 2^{-\ell(n)}. \tag{80}$$

Thus, for both $b \in \{0,1\}$,

$$\langle s_n^b | \, C_n(\varepsilon) \, | s_n^b \rangle < p(n) \cdot 2^{-\ell(n)} \tag{81}$$

infinitely often. Hence, $(s_n^b)_{n\in\mathbb{N}}$ is weakly ingenerable. $\square$

A.1 Preliminaries

We review here the basics of Kolmogorov complexity and Martin-Löf randomness as well as establish some extra notation pertaining to bit strings. We generally follow the textbook of Li and Vitányi [LV19].

A.1.1 More on Bit Strings and the cut_ℓ Map

We denote by $\{0,1\}^\infty$ be the set of all infinite sequences of bits. When we wish to emphasize that a sequence of 0's and 1's be interpreted as a bit string and not a number, we will write them as sequences of $\mathtt{0}$'s and $\mathtt{1}$'s.

For a Turing machine $\mathtt{T}$, we let $\langle \mathtt{T} \rangle \in \{0,1\}^*$ denote a reasonable binary encoding of $\mathtt{T}$. The specifics of this encoding are not needed for this work. Similarly, for any $n \in \mathbb{N}$ we let $\langle n \rangle \in \{0,1\}^*$ denote the shortest representation of n in binary. Here, we adopt the same convention as in [LV19] and represent the number 0 with the empty string and continue from there. Thus, $\langle 0 \rangle = \varepsilon$, $\langle 1 \rangle = \mathtt{0}$, and $\langle 3 \rangle = \mathtt{10}$. This yields a bijection between $\{0,1\}^*$ and $\mathbb{N}$ as well as the relation

$$|\langle n \rangle| = \lfloor \log(n+1) \rfloor. \tag{82}$$

We also define $\langle n \rangle_m$ to be the binary representation of n that is left-padded by 0's to be of length at least m, *e.g.* $\langle 5 \rangle_6 = \mathtt{000100}$.

For any string $x \in \{0,1\}^*$, we let ${}_{\mathrm{d}}x$ be the string which repeats every bit of x twice in succession. For example, ${}_{\mathrm{d}}\langle 5 \rangle = {}_{\mathrm{d}}\,\mathtt{101} = \mathtt{110011}$.

Let $s \in \{0,1\}^* \cup \{0,1\}^\infty$ be a finite or infinite string and let $n, m \in \mathbb{N}^+$ be two strictly positive integers. If $n \leq m \leq |s|$, we let $s_{[n,m]} \in \{0,1\}^{m-n+1}$ be the string composed precisely of the n-th to m-th bits, inclusively, of s. Otherwise, we set $s_{[n,m]} = \varepsilon$.

For any map $\ell : \mathbb{N} \to \mathbb{N}$, we denote the set of ℓ-sequences by $\{0,1\}^*_\ell$. We now define a map $\mathrm{cut}_\ell : \{0,1\}^\infty \to \{0,1\}^*_\ell$. Intuitively, cut_ℓ will "cut" an infinitely sequence $s \in \{0,1\}^\infty$ into finite strings s_λ of length $\ell(\lambda)$. The string s_0 will be first $\ell(0)$ bits of s, the string s_1 will be the next $\ell(1)$ bits, s_2 will be the next $\ell(2)$ bits, and so on and so forth. If the map ℓ is non-zero infinitely often, then cut_ℓ is a bijection and the inverse map $\mathrm{cut}_\ell^{-1} : \{0,1\}^*_\ell \to \{0,1\}^\infty$ is simply the infinite concatenation, in order, of all strings in a sequence $(s_\lambda)_{\lambda \in \mathbb{N}}$. More formally, let $L : \mathbb{N} \to \mathbb{N}$ be defined by $n \mapsto \sum_{i=0}^{n} \ell(i)$. Then, $\mathrm{cut}_\ell(s) = (s_n)_{n \in \mathbb{N}}$ where

$$s_n = \begin{cases} s_{[1,L(0)]} & \text{if} \quad n = 0 \\ s_{[1+L(n-1),L(n)]} & \text{else.} \end{cases} \tag{83}$$

A.1.2 Kolmogorov Complexity

We now move on to defining the Kolmogorov complexity of a string. This is also known as the *plain* Kolmogorov complexity when we wish to explicitly distinguish it from other variations of this definition, such as the prefix Kolmogorov complexity.[18]

Definition 52. Let $\mathtt{T}$ be a Turing machine. The Kolmogorov complexity of a string $x \in \{0,1\}^*$ with respect to the machine $\mathtt{T}$, denoted $K_\mathtt{T}(x)$, is defined by

$$K_\mathtt{T}(x) = \begin{cases} \infty & \text{if} \quad \mathtt{T}(y) \neq x \text{ for all } y \in \{0,1\}^* \\ \min\{|y| \ : \ \mathtt{T}(y) = x\} & \text{else.} \end{cases} \tag{84}$$

If $\mathtt{U}$ is a universal Turing machine, in the sense that $\mathtt{U}(\langle \mathtt{T} \rangle, x) = \mathtt{T}(x)$ for all strings $x \in \{0,1\}^*$ where $\mathtt{T}(x)$ halts, then there exists a constant $c \in \mathbb{N}$ such that $K_\mathtt{U}(x) \leq K_\mathtt{T}(x) + c$. Indeed, it suffices to take $c = |\langle \mathtt{T} \rangle|$. It follows that for any two universal Turing machines $\mathtt{U}$ and $\mathtt{U}'$ there exists a constant c such that $K_\mathtt{U}(x) \leq K_{\mathtt{U}'}(x) + c$ for all strings $x \in \{0,1\}^*$.

With the above in mind, we fix for the remainder of this section a universal Turing machine $\mathtt{U}$ and set $K = K_\mathtt{U}$. Up a to an additive constant, the particular choice of $\mathtt{U}$ does not matter.

As an immediate result, we note that there exists a constant c such that $K(x) \leq |x| + c$ for all strings $x \in \{0,1\}^*$. Indeed, let $\mathtt{Id}$ be the Turing machine which immediately halts. We then have that $\mathtt{U}(\langle Id \rangle, x) = x$ for all $x \in \{0,1\}^*$ and so $K(x) \leq |\langle \mathtt{Id} \rangle| + |x|$.

Finally, we also define the *conditional* Kolmogorov complexity. In short, we assume here that the Turing machine also has access to another string z when trying to compute x and the length of z is not counted in the resulting measure of complexity.

Definition 53. Let $\mathtt{T}$ be a Turing machine. The Kolmogorov complexity of a string $x \in \{0,1\}^*$ conditioned on $z \in \{0,1\}^*$ with respect to $\mathtt{T}$ is defined by

$$K_\mathtt{T}(x|z) = \begin{cases} \infty & \text{if} \quad \mathtt{T}({}_\mathrm{d}z, \mathtt{01}, y) \neq x \text{ for all } y \in \{0,1\}^* \\ \min\{|y| \ : \ \mathtt{T}({}_\mathrm{d}z, \mathtt{01}, y) = x\} & \text{else.} \end{cases} \tag{85}$$

Note that the double encoding of z followed by $\mathtt{01}$ allows the Turing machine to unambiguously distinguish between z and y.

We again fix a single universal Turing machine $\mathtt{U}$ and always measure the conditional Kolmogorov complexity with respect to this machine.

[18]Note that all our references on this topic use C for the plain Kolmogorov complexity and reserve K for the prefix Kolmogorov complexity. We use K for the Kolmogorov complexity as we typically use C to denote circuits.

A.1.3 Martin-Löf Random Sequences

A very succinct characterization of Martin-Löf randomness can be given in terms of prefix Kolmogorov complexity. We take this characterization as our definition. We do not formally define prefix Kolmogorov complexity as we will not directly be using this characterization, but it roughly corresponds to restricting the plain Kolmogorov complexity to only consider Turing machines whose behaviours are completely characterized by their actions on a prefix set.

Definition 54 ([LV19, Theorem 3.5.1])**.** A string $w \in \{0,1\}^\infty$ is Martin-Löf random if there exists a constant c such that the prefix Kolmogorov complexity of $w_{[1,n]}$ is at least $n - c$ for all $n \in \mathbb{N}$.

At a high level, this characterization captures the idea that there should be a limit on how much the prefixes of a Martin-Löf random sequence can be compressed. For our needs, the following two theorems will be sufficient. The first, due to Miller and Yu [MY08] (although we use a restricted version of the formulation given in [LV19, Theorem 2.5.4]) is expressed in terms of the conditional plain Kolmogorov complexity. The second is due to Schnorr [Sch71] and is based on the concept of computable martingales.[19]

Theorem 55. For all Martin-Löf random $w \in \{0,1\}^\infty$, there exists a $c \in \mathbb{N}$ such that for all $n \in \mathbb{N}$

$$K(w_{[1:n]}|\langle n\rangle) \geq n - 2\log(n) - c. \tag{86}$$

Theorem 56. Let $f : \{0,1\}^* \to \mathbb{N}$ be a computable map such that for all $x \in \{0,1\}^*$ we have that

$$f(x) = \frac{f(x,0) + f(x,1)}{2}. \tag{87}$$

Any $w \in \{0,1\}^\infty$ such that $\limsup_{n\to\infty} f(w_{[1:n]}) = \infty$ is not Martin-Löf random.

A.2 Ingenerable Sequences Need Not be Martin-Löf Random

We prove in this section Theorem 59 which states that, under the action of the cut_ℓ^{-1} bijection for a computable map ℓ, ingenerable sequences may not yield Martin-Löf random sequences. The core of our argument is that we can find ingenerable sequences such that their concatenation yields an infinite sequence of bits with 0's at infinitely many locations which can be computed. This is sufficient to ensure non-Martin-Löf randomness as it will allow us to construct a computable martingale, in the sense of Theorem 56, who's value will grow to infinity.

In practice, we will use the following corollary of Theorem 56.

Corollary 57. Let $L : \mathbb{N} \to \mathbb{N}^+$ be a computable strictly monotone map. If a string $s \in \{0,1\}^\infty$ is such that $s_{[L(i):L(i)]} = 0$ for all $i \in \mathbb{N}$, then s is not Martin-Löf random.

Proof. Consider a Turing `T` machine which, on input of a string $x \in \{0,1\}^*$, implements the following algorithm:

1. Initialize a counter v with value 1 and a counter i with value 0.
2. Compute $L(i)$. If $L(i) > |x|$, then output the value of v and halt. Else:
   (a) Check if $x_{[L(i),L(i)]}$ is 0. If it is, double the value of v. If it is not, set v to be 0.

[19] Slightly relaxing the conditions on f in this theorem, namely only requiring it to be *weakly computable* and letting its codomain be $\mathbb{R}_0^+$, actually yields a characterization of non-Martin-Löf randomness.

(b) Increment i by setting it to $i+1$ and return to step 2.

Note that T halts on every input since L is strictly monotone. Let $f : \{0,1\}^* \to \mathbb{N}$ be the function computed by T and note that $f(x) = \frac{f(x,0)+f(x,1)}{2}$ for all strings $x \in \{0,1\}^*$. By Theorem 56, it suffices to show that $\lim_{n\to\infty} f(s_{0:n}) = \infty$ to obtain that s is not Martin-Löf random. To do this, we simply note that

$$f(s_{[1,n]}) = 2^{|L(\mathbb{N})\cap[n]|} \tag{88}$$

and that since $|L(\mathbb{N})|$ is infinite, this value tends to infinity as n grows. Indeed, consider the value of the register v maintained by the Turing machine described above when it is run on $s_{[1:n]}$. Step 2(a) will be executed exactly $|L(\mathbb{N}) \cap [n]|$ times. By hypothesis, $s_{[L(i),L(i)]}$ is always 0 and thus the value of v is doubled every time. This yields the desired result. □

At this point, it essentially suffices to argue that there are ingenerable sequences with infinitely many zeros at precisely computable locations.

Lemma 58. Let $(s_\lambda)_{\lambda\in\mathbb{N}}$ be an ingenerable ℓ-sequence. For all $\lambda \in \mathbb{N}$, let s'_λ be s_λ, except with the first bit replaced by 0, unless $s_\lambda = \varepsilon$ in which case no change is made. Then the sequence $(s'_\lambda)_{\lambda\in\mathbb{N}}$ is also ingenerable.

Proof. Let $(C'_\lambda)_{\lambda\in\mathbb{N}}$ be a uniform $(0,\ell)$-circuit family. (If no such family exists, such as if ℓ is uncomputable, then the proof is complete.) For all $\lambda \in \mathbb{N}$, let C_λ be the circuit which first runs the circuit C'_λ and then replaces the first qubit with the maximally mixed state, unless C'_λ is a $(0,0)$-circuit in which case $C_\lambda = C'_\lambda$. Clearly, $(C_\lambda)_{\lambda\in\mathbb{N}}$ is also a uniform $(0,\ell)$-circuit family. Note that

$$\frac{1}{2} \langle s'_\lambda | \, C'_\lambda(\varepsilon) \, | s_\lambda \rangle \leq \langle s_\lambda | \, C_\lambda(\varepsilon) \, | s_\lambda \rangle \tag{89}$$

since, conditioned on C_λ correctly outputting the last $\ell(\lambda) - 1$ bits of s_λ, C'_λ has a probability $\frac{1}{2}$ of outputting s'_λ. Note that if $\ell(\lambda) = 0$, the inequality still holds as the left-hand side is $\frac{1}{2}$ and the right hand side is 1. Now, since $(s_\lambda)_{\lambda\in\mathbb{N}}$ is ingenerable, there exists a polynomial p such that

$$\langle s_\lambda | \, C_\lambda(\varepsilon) \, | s_\lambda \rangle \leq p(\lambda) \cdot 2^{-\ell(\lambda)} \tag{90}$$

for all sufficiently large values of λ. Combining this with the previous inequality yields the desired result as $2p$ is a polynomial. □

We now pull everything together to state and prove the theorem.

Theorem 59. Let $\ell : \mathbb{N} \to \mathbb{N}$ be a computable function which is nonzero infinitely often. Then, there exists an ingenerable ℓ-sequence $(s_\lambda)_{\lambda\in\mathbb{N}}$ such that $\mathrm{cut}_\ell^{-1}((s_\lambda)_{\lambda\in\mathbb{N}})$ is not Martin-Löf random.

Proof. By Theorem 9, there exists an ingenerable ℓ-sequence $(s_\lambda)_{\lambda\in\mathbb{N}}$ and by Lemma 58 we can assume that the first bit of every non-empty s_λ in this sequence is 0. Let $N = \{n_0, n_1, n_2, \ldots\} \subseteq \mathbb{N}$ be the set of values on which ℓ is nonzero. We index the values of N such that $n_j < n_{j+1}$ for all $j \in \mathbb{N}$. Now, define the map $L : \mathbb{N} \to \mathbb{N}^+$ by

$$i \mapsto 1 + \sum_{j=0}^{i-1} \ell(n_j). \tag{91}$$

and note that it is computable and strictly increasing. By construction, the $L(i)$-th bit of $\mathrm{cut}_\ell^{-1}(s_\lambda)$ is 0 for all $i \in \mathbb{N}$. Indeed, $L(i)$ identifies the location where the first bit of s_{n_i} will be situated in the infinite string $\mathrm{cut}_\ell^{-1}((s_\lambda)_{\lambda\in\mathbb{N}})$. Thus, by Corollary 57, this string is not Martin-Löf random. □

A.3 Martin-Löf Random Sequences give Weakly Ingenerable Sequences

The key observation in this section is that if we already given the description of a circuit C which generates a string s with at least some probability p, *i.e.:* $\langle s| C(\varepsilon) |s\rangle \geq p$, then we can describe s with at most $\approx \log(p^{-1})$ extra bits. Indeed, there are at most p^{-1} strings satisfying the previous inequality and and we can identify s by by specifying it's location, in lexicographic order, in the set of such strings. If p is large enough, this will be shorter than $|s|$.

Using this idea, we can show that if $\mathrm{cut}_\ell(s) = (s_\lambda)_{\lambda\in\mathbb{N}}$ is not weakly ingenerable if ℓ is a polynomial, as in that case all but some initial set of segments of s can be compressed by the above idea. Moreover, the C circuits can in fact be taken from a uniform family, which is to say that they can all be generate by a single Turing machine and thus they only have a constant contribution to the Kolmogorov complexity. This, and a few extra technical considerations, is enough to show that s is not Martin-Löf random. Taking the contrapositive yields the following theorem.

Theorem 60. Let $s \in \{0,1\}^\infty$ be a Martin-Löf random sequence and let $\ell : \mathbb{N} \to \mathbb{N}$ be a polynomial. Then, $\mathrm{cut}_\ell(s)$ is weakly ingenerable.

Proof. Assume that $\ell(\lambda)$ is not constant. If it was, then $\mathrm{cut}_\ell(s)$ would be trivially ingenerable, and hence weakly ingenerable, by virtue of $\ell \notin \omega(\log)$ and the proof would be done. Note that since ℓ is never negative, this implies that $\lim_{\lambda\to\infty} \ell(\lambda) = \infty$.

Aiming for a contradiction, assume that the ℓ-sequence $\mathrm{cut}_\ell(s) = (s_\lambda)_{\lambda\in\mathbb{N}}$ is not weakly ingenerable. Thus, there exists a uniform family of $(0,\ell)$-circuits $(C_\lambda)_{\lambda\in\mathbb{N}}$ such that

$$\langle s_\lambda| C_\lambda(\varepsilon) |s_\lambda\rangle \geq (p(\lambda)+2)\cdot 2^{-\ell(\lambda)} \quad \text{where} \quad p(\lambda) = 2\ell(\lambda+1)^3 \tag{92}$$

for all but finitely many values of λ. Let $\mathtt{C}$ be a Turing machine which generates this circuit family.

Our goal is now to describe a Turing machine $\mathtt{GEN}$ and strings $a_n \in \{0,1\}^*$ such that for all constants c, there exists an n_c such that

$$\mathtt{GEN}(_\mathrm{d}\langle n_c\rangle, 01, a_{n_c}) = s_{[1,n_c]} \quad \text{and} \quad |a_{n_c}| < n_c - 2\log(n_c) - c \tag{93}$$

Taking the contrapositive of Theorem 55, this is sufficient to conclude that s is not Martin-Löf random, yielding our contradiction.

We construct $\mathtt{GEN}$ from a few subroutines, which we will also describe as Turing machines. First, let $\mathtt{SIM}_\mathtt{C}$ be a Turing machine which, on input of $\langle\lambda\rangle$ and a string $y \in \{0,1\}^{\ell(\lambda)}$, computes the value of $\langle y| C_\lambda(\varepsilon) |y\rangle$ within an additive error of $2^{-\ell(\lambda)}$. Next, let $\mathtt{SET}_\mathtt{C}$ be a Turing machine which on input of $\langle\lambda\rangle$ implements the following algorithm:

1. Initialize an empty set $\mathcal{S}$.
2. Iterating over all strings $y \in \{0,1\}^{\ell(\lambda)}$, do the following:
   (a) Run $\mathtt{SIM}_\mathtt{C}(_\mathrm{d}\langle\lambda\rangle, \mathtt{01}, y)$.
   (b) If this determines that $\langle y| C_\lambda(\varepsilon) |y\rangle \geq (p(\lambda)+1)\cdot 2^{-\ell(\lambda)}$, add y to $\mathcal{S}$.
3. Output a description of the set $\mathcal{S}$ and halt.

Let's establish a bit of notation for the sets $\mathcal{S}$ computed by $\mathtt{SET}_\mathtt{C}$ and examine some of their properties. This will also allow us to define the strings a_n before moving on to describing $\mathtt{GEN}$.

For all $\lambda \in \mathbb{N}$, let $\mathcal{S}_\lambda$ be the set ultimately computed by $\mathtt{SET}_\mathtt{C}(\langle\lambda\rangle)$. Note that, by the additive error bound we imposed on $\mathtt{SIM}_\mathtt{C}$, this set contains every y satisfying $\langle y| C_\lambda(\varepsilon) |y\rangle \geq (p(\lambda)+2)\cdot 2^{-\ell(\lambda)}$

and none satisfying $\langle y| C_\lambda(\varepsilon) |y\rangle < p(\lambda) \cdot 2^{-\ell(\lambda)}$. This implies that for every λ such that $p(\lambda) \neq 0$, the set $\mathcal{S}_\lambda$ contains at most $p(\lambda)^{-1} \cdot 2^{\ell(\lambda)}$ elements and that $s_\lambda \in \mathcal{S}_\lambda$ for all but finitely many values of λ. If $s_\lambda \in \mathcal{S}_\lambda$, we say that s_λ is compressible. If s_λ is compressible, we let $r_\lambda \in \mathbb{N}$ denote it's position, in the lexicographic order, among the elements of $\mathcal{S}_\lambda$. We let n_λ be the maximal number of bits required to express r_λ. If $p(\lambda) \neq 0$, then

$$n_\lambda = |\langle |\mathcal{S}_\lambda| \rangle| \leq \log(|\mathcal{S}_\lambda| + 1) \leq \log(2|\mathcal{S}_\lambda|) \leq \log\left(2 \cdot \frac{2^{\ell(\lambda)}}{p(\lambda)}\right) = \ell(\lambda) - 3\log(\ell(\lambda+1)). \tag{94}$$

We let

$$\tilde{\lambda} = \max\left\{\{\lambda \ : \ s_\lambda \notin \mathcal{S}_\lambda\} \cup \{\lambda \ : \ p(\lambda) = 0\}\right\} \tag{95}$$

be the largest integer such that $s_{\tilde{\lambda}}$ is not compressible or that $\ell(\tilde{\lambda}) = 0$. Recall that ℓ tends to infinity and so such a $\tilde{\lambda}$ exists. If $\lambda > \tilde{\lambda}$, then $s_\lambda \in \mathcal{S}_\lambda$ and it takes at most $\ell(\lambda) - 2\log(\ell(\lambda+1))$ bits to give its location in the set.

We now define a_n for all $n \in \mathbb{N}$. Let $L : \mathbb{N} \to \mathbb{N}$ be defined by $\lambda \mapsto \sum_{i=0}^{\lambda} \ell(i)$ and let $\lambda_n \in \mathbb{N}$ be the largest integer such that $L(\lambda_n) \leq n$, if such an integer exists. In other words, λ_n is the number of complete strings from the sequence $(s_\lambda)_{\lambda \in \mathbb{N}}$ which appear in $s_{[1:n]}$. Then,

$$a_n = \begin{cases} s_{[1,n]} & \text{if} \quad n \leq L(\tilde{\lambda}) \\ \left(s_{[1,L(\tilde{\lambda})]}, \langle r_{\tilde{\lambda}+1} \rangle_{n_{\tilde{\lambda}+1}}, \langle r_{\tilde{\lambda}+2} \rangle_{n_{\tilde{\lambda}+2}}, \ldots, \langle r_{\lambda_n} \rangle_{n_{\lambda_n}}, s_{[L(\lambda_n)+1,n]}\right) & \text{else.} \end{cases} \tag{96}$$

In general (*i.e.*: the second case in the above equation), a_n includes the following information:

- A prefix $s_{[1,L(\tilde{\lambda})]}$ of $s_{[1,n]}$ which may not be compressible by our method.
- For every s_λ which appears in whole in $s_{[1,n]}$ and which can be compressed by our method, a_n includes the value of r_λ in binary and padded to a predictable length.
- The suffix $s_{[L(\lambda_n)+1,n]}$, representing the incomplete part of s_{λ_n+1} which is present in $s_{[1,n]}$.

For all n such that $\lambda_n > \tilde{\lambda}$ we have that

$$\begin{aligned} |a_n| &= L(\tilde{\lambda}) + \left(\sum_{\lambda=\tilde{\lambda}+1}^{\lambda_n} n_\lambda\right) + (n - L(\lambda_n)) \\ &\leq L(\tilde{\lambda}) + \left(\sum_{\lambda=\tilde{\lambda}+1}^{\lambda_n} \ell(\lambda) - 3\log(\ell(\lambda+1))\right) + (n - L(\lambda_n)) \\ &\leq n - \sum_{\lambda=\tilde{\lambda}+1}^{\lambda_n} 3\log(\ell(\lambda+1)) \\ &= n - \sum_{\lambda=\tilde{\lambda}+2}^{\lambda_n+1} 3\log(\ell(\lambda)) \\ &\leq n - 3\log\left(\sum_{\lambda=\tilde{\lambda}+2}^{\lambda_n+1} \ell(\lambda)\right) \\ &= n - 3\log\left(L(\lambda_n+1) - L(\tilde{\lambda}+1)\right) \end{aligned} \tag{97}$$

Using the facts that $n < L(\lambda_n + 1)$ and that if $x \in \mathbb{N}$ and $y \geq x + 1$ then $\log(y - x) \geq \log(y) - x$, we obtain

$$\begin{aligned} |a_n| &\leq n - 3\log(L(\lambda_n + 1)) + 3L(\tilde{\lambda} + 1) \\ &\leq n - 3\log(n) + 3L(\tilde{\lambda} + 1) \\ &= n - 2\log(n) + 3L(\tilde{\lambda} + 1) - \log(n) \end{aligned} \tag{98}$$

Since $3L(\tilde{\lambda} + 1)$ is a constant, we have that

$$\lim_{n \to \infty} 3L(\tilde{\lambda} + 1) - \log(n) = -\infty \tag{99}$$

and so for every constant c there exists an $n_c \in \mathbb{N}$ such that $|a_{n_c}| < n_c - 2\log(n_c) - c$. □

B Supplementary Proofs

B.1 Proof of Lemma 11

Before proving Lemma 11, we highlight a small technical fact. If a subset $\mathcal{S}$ of a finite set $\mathcal{X}$ is constructed by iterating over all elements $x \in \mathcal{X}$ and including this element in $\mathcal{S}$ with probability p_x, independently of all other choices, then the expected size of $\mathcal{S}$ is $\sum_{x \in \mathcal{X}} p_x$.

Fact 61. Let $\mathcal{X}$ be a finite set and, for every $x \in \mathcal{X}$, let $p_x \in [0, 1]$ be a real number. Let $\mathcal{S}$ be a random variable distributed on $\mathcal{P}(\mathcal{X})$ such that, for all $\mathcal{Y} \in \mathcal{P}(\mathcal{X})$

$$\Pr[\mathcal{S} = \mathcal{Y}] = \left(\prod_{x \in \mathcal{Y}} p_x\right) \cdot \left(\prod_{x \in \mathcal{Y} \setminus \mathcal{S}} 1 - p_x\right). \tag{100}$$

Then, $\mathbb{E}\,|\mathcal{S}| = \sum_{x \in \mathcal{X}} p_x$.

Proof. For any $\mathcal{Y} \in \mathcal{X}$, let $\delta_{\mathcal{Y}} : \mathcal{X} \to \{0, 1\}$ be the characteristic function for the set $\mathcal{Y}$. We can then compute $\mathbb{E}\,|\mathcal{S}|$ as

$$\sum_{\mathcal{Y} \in \mathcal{P}(\mathcal{X})} |\mathcal{Y}| \cdot \Pr[\mathcal{S} = \mathcal{Y}] = \sum_{\mathcal{Y} \in \mathcal{P}(\mathcal{X})} \sum_{x \in \mathcal{X}} \delta_{\mathcal{Y}}(x) \cdot \Pr[\mathcal{S} = \mathcal{Y}] = \sum_{x \in \mathcal{X}} \sum_{\mathcal{Y} \in \mathcal{P}(\mathcal{X})} \delta_{\mathcal{Y}}(x) \cdot \Pr[\mathcal{S} = \mathcal{Y}] \tag{101}$$

which is precisely $\sum_{x \in \mathcal{X}} p_x$, the desired result, as $\sum_{\mathcal{Y} \in \mathcal{P}(\mathcal{X})} \delta_{\mathcal{Y}}(x) \cdot \Pr[\mathcal{S} = \mathcal{Y}] = p_x$. □

We can now prove the lemma.

Proof of Lemma 11. Let $\mathcal{S}$ be the random variable distributed on $\mathcal{P}(\{0, 1\}^n)$ which models the contents of the set maintained by the circuit $\tilde{C}$ once it terminates step 2. Note that

$$\Pr[\mathcal{S} = \mathcal{X}] = \left(\prod_{x \in \mathcal{X}} \langle 1|\, C(x)\, |1\rangle\right) \left(\prod_{x \in \{0,1\}^n \setminus \mathcal{X}} 1 - \langle 1|\, C_x(x)\, |1\rangle\right). \tag{102}$$

We see that

$$\langle s|\, \tilde{C}(s)\, |s\rangle \geq \sum_{\substack{\mathcal{X} \in \mathcal{P}(\{0,1\}^n) \\ s \in \mathcal{X}}} \frac{1}{|\mathcal{X}|} \cdot \Pr[\mathcal{S} = \mathcal{X}] \tag{103}$$

where the inequality is obtained by neglecting the case where $\mathcal{S}$ is empty.

Next, let $\mathcal{S}'$ be the random variable distributed on $\mathcal{P}\left(\{0,1\}^n \setminus \{s\}\right)$ such that

$$\Pr\left[\mathcal{S}' = \mathcal{X}\right] = \left(\prod_{x\in\mathcal{X}} \langle 1| C(x) |1\rangle\right)\left(\prod_{x\in(\{0,1\}^n\setminus\{s\})\setminus\mathcal{X}} 1 - \langle 1| C_x(x) |1\rangle\right) \tag{104}$$

and note that if $s \in \mathcal{X}$ then $\Pr[\mathcal{S} = \mathcal{X}] = \langle 1| C(s) |1\rangle \cdot \Pr[\mathcal{S}' = \mathcal{X} \setminus \{s\}]$ for any $\mathcal{X} \in \mathcal{P}(\{0,1\}^n)$. We then have that

$$\begin{aligned}
\sum_{\substack{\mathcal{X}\in\mathcal{P}(\{0,1\}^n)\\ s\in\mathcal{X}}} \frac{1}{|\mathcal{X}|} \cdot \Pr\left[\mathcal{S} = \mathcal{X}\right] &= \sum_{\substack{\mathcal{X}\in\mathcal{P}(\{0,1\}^n)\\ s\in\mathcal{X}}} \frac{1}{1 + |\mathcal{X} \setminus \{s\}|} \cdot \Pr\left[\mathcal{S} = \mathcal{X}\right] \\
&= \sum_{\substack{\mathcal{X}\in\mathcal{P}(\{0,1\}^n)\\ s\in\mathcal{X}}} \frac{\langle s| C(s) |s\rangle}{1 + |\mathcal{X} \setminus \{s\}|} \cdot \Pr\left[\mathcal{S}' = \mathcal{X} \setminus \{s\}\right] \\
&= \sum_{\mathcal{X}'\in\mathfrak{P}(\{0,1\}^n\setminus\{s\})} \frac{\langle s| C(s) |s\rangle}{1 + |\mathcal{X}'|} \cdot \Pr\left[\mathcal{S}' = \mathcal{X}'\right] \qquad (105)\\
&= \langle s| C(s) |s\rangle \cdot \mathbb{E}\,\frac{1}{1 + |\mathcal{S}'|} \\
&\geq \langle s| C(s) |s\rangle \cdot \frac{1}{1 + \mathbb{E}\,|\mathcal{S}'|} \\
&= \langle s| C(s) |s\rangle \cdot \frac{1}{1 + \sum_{x\in\{0,1\}^n\setminus\{s\}} \langle 1| C(x) |1\rangle}
\end{aligned}$$

where the inequality is obtained by Jensen's inequality and the expectation is evaluated by Fact 61. Combining this with Equation (103) yields the desired result. □

References

[Aar05] S. Aaronson. Limitations of quantum advice and one-way communication. *Theory of Computing*, 1(1):1–28, 2005.
doi:10.4086/toc.2005.v001a001

[Aar09] S. Aaronson. Quantum copy-protection and quantum money. In *24th Annual Conference on Computational Complexity (CCC 2009)*, pages 229–242, 2009.
doi:10.1109/CCC.2009.42

[AB09] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[AC02] M. Adcock and R. Cleve. A quantum Goldreich-Levin theorem with cryptographic applications. In *19th Annual Symposium on Theoretical Aspects of Computer Science — STACS 2002*, pages 323–334, 2002.
doi:10.1007/3-540-45841-7_26

[AKL23] P. Ananth, F. Kaleoglu, and Q. Liu. Cloning games: A general framework for unclonable primtives. 2023.
arXiv:2302.01874

[ALL+21] S. Aaronson, J. Liu, Q. Liu, M. Zhandry, and R. Zhang. New approaches for quantum copy-protection. In *Advances in Cryptology — CRYPTO 2021*, volume 1, pages 526–555, 2021.
doi:10.1007/978-3-030-84242-0_19

[ALP21] P. Ananth and R. L. La Placa. Secure software leasing. In *Advances in Cryptology — EUROCRYPT 2021*, volume 2, pages 501–530, 2021.
doi:10.1007/978-3-030-77886-6_17

[BB84] C. H. Bennett and G. Brassard. Quantum cryptography: Public key distribution and coin tossing. In *International Conference on Computers, Systems and Signal Processing*, pages 175–179, 1984.

[BDE+98] D. Bruß *et al.* Optimal universal and state-dependent quantum cloning. *Physical Review A*, 57(4):2368, 1998.
doi:10.1103/PhysRevA.57.2368

[Bel02] M. Bellare. A note on negligible functions. *Journal of Cryptology*, 15(4):271–284, 2002.
doi:10.1007/s00145-002-0116-x

[BGT21] A. Bouland and T. Giurgică-Tiron. Efficient universal quantum compilation: An inverse-free Solovay-Kitaev algorithm. 2021.
arXiv:2112.02040v1 [quant-ph]

[BH96] V. Bužek and M. Hillery. Quantum copying: Beyond the no-cloning theorem. *Physical Review A*, 54(3):1844–1852, 1996.
doi:10.1103/PhysRevA.54.1844

[BJL+21] A. Broadbent, S. Jeffery, S. Lord, S. Podder, and A. Sundaram. Secure software leasing without assumptions. In *Theory of Cryptography (TCC 2021)*, volume 1, pages 90–120, 2021.
doi:10.1007/978-3-030-90459-3_4

[BL20] A. Broadbent and S. Lord. Uncloneable quantum encryption via oracles. In *15th Conference on the Theory of Quantum Computation, Communication, and Cryptography (TQC 2020)*, article no. 4, 2020.
doi:10.4230/LIPIcs.TQC.2020.4

[Bra85] G. Brassard. Crusade for a better notation. *ACM SIGACT News*, 17(1):60–64, 1985.
doi:10.1145/382250.382808

[BV97] E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997.
doi:10.1137/S0097539796300921

[CLLZ21] A. Coladangelo, J. Liu, Q. Liu, and M. Zhandry. Hidden cosets and applications to unclonable cryptography. In *Advances in Cryptology — CRYPTO 2021*, pages 556–584, 2021.
doi:10.1007/978-3-030-84242-0_20

[CMP20] A. Coladangelo, C. Majenz, and A. Poremba. Quantum copy-protection of compute-and-compare programs in the quantum random oracle model. 2020.
arXiv:2009.13865 [quant-ph]

[Die82] D. Dieks. Communication by EPR devices. *Physics Letters A*, 92(6):271–272, 1982.
doi:10.1016/0375-9601(82)90084-6

[DN06] C. M. Dawson and M. A. Nielsen. The Solovay-Kitaev algorithm. *Quantum Information & Computation*, 6(1):81–95, 2006.
doi:10.26421/QIC6.1-6

[EPR35] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Physical Review Letters*, 47(10):777–780, 1935.
doi:10.1103/physrev.47.777

[Got03] D. Gottesman. Uncloneable encryption. *Quantum Information & Computation*, 3(6):581–602, 2003.
doi:10.26421/QIC3.6-2

[Her82] N. Herbert. FLASH — A superluminal communicator based upon a new kind of quantum measurement. *Foundations of Physics*, 12(12):1171–1179, 1982.
doi:10.1007/BF00729622

[JJUW11] R. Jain, Z. Ji, S. Upadhyay, and J. Watrous. QIP = PSPACE. *Journal of the ACM*, 58(6):30, 2011.
doi:10.1145/2049697.2049704

[Kit97] A. Y. Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191–1249, 1997.
doi:10.1070/RM1997v052n06ABEH002155

[Knu76] D. E. Knuth. Big omicron and big omega and big theta. *ACM SIGACT News*, 8(2):18–24, 1976.
doi:10.1145/1008328.1008329

[KT22] S. Kundu and E. Y.-Z. Tan. Device-independent uncloneable encryption. 2022.
arXiv:2210.01058 [quant-ph]

[LFM$^+$23] M. Lemus, R. Faleiro, P. Mateus, N. Paunković, and A. Souto. Quantum Kolmogorov complexity and quantum correlations in deterministic-control Turing machines. 2023.
arXiv:2305.14252 [quant-ph]

[LV19] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications.* Springer, fourth edition, 2019.
doi:10.1007/978-3-030-11298-1

[Mae13] R. Maes. *Physically Unclonable Functions.* Springer, 2013.
doi:10.1007/978-3-642-41395-7

[MVW13] A. Molina, T. Vidick, and J. Watrous. Optimal counterfeiting attacks and generalizations for Wiesner's quantum money. In *Theory of Quantum Computation, Communication, and Cryptography (TQC 2012)*, pages 45–64, 2013.
doi:10.1007/978-3-642-35656-8_4

[MW05] C. Marriott and J. Watrous. Quantum Arthur–Merlin games. *Computational Complexity*, 14(2):122–152, 2005.
doi:10.1007/s00037-005-0194-x

[MY08] J. S. Miller and L. Yu. On initial segment complexity and degrees of randomness. *Transactions of the American Mathematical Society*, 360(6):3193–3210, 2008.
doi:10.1090/S0002-9947-08-04395-X

[MY23] T. Metger and H. Yuen. stateQIP = statePSPACE. 2023.
arXiv:2301.07730 [quant-ph]

[NC10] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 10th anniversary edition, 2010.
doi:10.1017/CBO9780511976667

[NW06] T. Neary and D. Woods. Small fast universal turing machines. *Theoretical Computer Science*, 362(1):171–195, 2006.
doi:10.1016/j.tcs.2006.06.002

[NY04] H. Nishimura and T. Yamakami. Polynomial time quantum computation with advice. *Information Processing Letters*, 90(4):195–204, 2004.
doi:10.1016/j.ipl.2004.02.005

[OED22] Clonable, adj. In *Oxford English Dictionary Online*. 2022.
www.oed.com/view/Entry/272601

[Ort18] J. Ortigoso. Twelve years before the quantum no-cloning theorem. *American Journal of Physics*, 86(3):201–205, 2018.
doi:10.1119/1.5021356

[Par70] J. L. Park. The concept of transition in quantum mechanics. *Foundations of Physics*, 1(1):23–33, 1970.
doi:10.1007/BF00708652

[RY21] G. Rosenthal and H. Yuen. Interactive proofs for synthesizing quantum states and unitaries. 2021.
arXiv:2108.07192 [quant-ph]

[Sch71] C. P. Schnorr. A unified approach to the definition of random sequences. *Mathematical Systems Theory*, 5(3):246–258, 1971.
doi:10.1007/BF01694181

[TFKW13] M. Tomamichel, S. Fehr, J. Kaniewski, and S. Wehner. A monogamy-of-entanglement game with applications to device-independent quantum cryptography. *New Journal of Physics*, 15(10):103002, 2013.
doi:10.1088/1367-2630/15/10/103002

[Wat09] J. Watrous. Quantum computational complexity. In *Encyclopedia of Complexity and Systems Science*, pages 7174–7201. Springer, 2009.
doi:10.1007/978-3-642-27737-5_428-3

[Wat18] J. Watrous. *The Theory of Quantum Information*. Cambridge University Press, 2018.
doi:10.1017/9781316848142

[Wer98] R. F. Werner. Optimal cloning of pure states. *Physical Review A*, 58(3):1827, 1998.
doi:10.1103/PhysRevA.58.1827

[Wie83] S. Wiesner. Conjugate coding. *ACM SIGACT News*, 15(1):78–88, 1983. doi:10.1145/1008908.1008920

[WZ82] W. K. Wootters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, 299:802–803, 1982. doi:10.1038/299802a0

[Zha12] M. Zhandry. How to construct quantum random functions. In *2012 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2012)*, pages 679–687, 2012. doi:10.1109/FOCS.2012.37