Building Unclonable Cryptography: A Tale of Two No-cloning Paradigms

Ghada Almashaqbeh[1] and Rohit Chatterjee[2]

[1] University of Connecticut, ghada@uconn.edu
[2] Stony Brook University, rochatterjee@cs.stonybrook.edu

Abstract. Unclonable cryptography builds primitives that enjoy some form of unclonability, such as quantum money, software copy protection, and bounded execution programs. These are impossible in the classical model as classical data is inherently clonable. Quantum computing, with its no-cloning principle, offers a solution. However, it is not enough to realize bounded execution programs; these require one-time memory devices that self-destruct after a single data retrieval query. Very recently, a new no-cloning technology has been introduced [Eurocrypt'22], showing that unclonable polymers—proteins—can be used to build bounded-query memory devices and unclonable cryptographic applications.

In this paper, we investigate the relation between these two technologies; whether one can replace the other, or complement each other such that combining them brings the best of both worlds. Towards this goal, we review the quantum and unclonable polymer models, and existing unclonable cryptographic primitives. Then, we discuss whether these primitives can be built using the other technology, and show alternative constructions and notions when possible. We also offer insights and remarks for the road ahead. We believe that this study will contribute in advancing the field of unclonable cryptography on two fronts: developing new primitives, and realizing existing ones using new constructions.

1 Introduction

Unclonable cryptography is an emerging field building cryptographic primitives that enjoy some form of unclonability. Examples include quantum money [2], software copy protection [1], secure software leasing [8], bounded execution programs [17], and unclonable encryption [18]. Clearly, unclonable cryptography is impossible in the classical model; digital data can be copied and processed as many times as desired. This led to exploring new computing models to obtain the unclonability feature.

A no-cloning technology. Quantum computing, with its no-cloning principle, offers a solution in which quantum states that cannot be copied are used to encode information, e.g., a decryption key or a ciphertext. This means that only one party can process a quantum state at a time. Hence, if an adversary steals this state, the honest recipient will know that something went wrong. This also means that once the state is returned back to the original owner, even the honest recipient cannot execute the functionality anymore.

Bounded execution software. Unfortunately, quantum computing is not enough to realize bounded execution programs known as one or k-time program [17]. These are programs that can be executed only over one, or up to k, distinct inputs. Goldwasser et al. [17] showed that a one-time memory device—a memory that stores two secrets and allows only one of them to be retrieved—is needed to enforce bounded execution. They assumed the existence of such devices while outlining the use of secure hardware tokens as a general way to build them. This comes under the assumption that these tokens resist side-channel attacks and reverse engineering, which is problematic [14]. Given the quantum no-cloning principle, and that measuring a state is destructive, one may hope that the quantum model can realize k-time programs. This was ruled out in [10] which showed that k-time quantum programs are impossible without one-time memory devices.

An alternative no-cloning technology. Very recently, a new unclonability technology has been introduced [4]. It uses unclonable polymers, namely proteins, to build real-world bounded-query memory devices. Proteins, similar to DNA, can be used to store digital data but they enjoy unique features that DNA does not have: first, there is no way to clone a protein that is given in small amounts, and second, the only known way to sequence proteins and retrieve the digital data is using mass spectrometry that destructs the protein sample permanently. Based on that, Almashaqbeh et al. developed consumable tokens—(biological) memory devices that self-destruct after a few data retrieval attempts, and showed how to use them in building unclonable cryptographic applications.

The path ahead. With these two different no-cloning technologies, several questions arise on whether one can replace the other, or whether combining them may lead to a stronger notion achieving the best of both worlds. We believe that answering these questions will contribute in advancing the field of unclonable cryptography on two fronts: developing new unclonable primitives, and realizing existing ones using new constructions.[3]

Contributions. In this paper, we review each of the quantum and unclonable polymer models with a focus on the meaning (in terms of capabilities and limitations) of their no-cloning principles. This comparative approach not only helps with understanding these models, but also delineates the path towards their use in unclonable cryptography. We then discuss several unclonable primitives built so far (where naturally, most of these are in the well-established quantum setting), while showing new constructions in the other no-cloning model with alternative security notions (if needed) or reasoning why it is impossible to achieve that. This is in addition to studying whether these functionalities can be realized using bounded-execution programs. In general, we find that the power gap between the honest party and the adversary (in terms of number of queries) in the unclonable polymer setting limits the applications that can be built using

[3] Physical unclonable functions (PUFs) [21] are another example of a no-cloning technology but a limited one. PUFs are not storage devices; they are physical (uncontrollable) sources of randomness, and they do not support bounded query or self-destruction. Thus, we consider PUFs outside the scope of this work.

this technology. That is, it either makes some applications impossible, or allows realizing them under weaker security notions. Lastly, we conclude with insights and takeaways advocating for a hybrid model that combines the two no-cloning technologies.

Related work. A recent work [22] presents a review of unclonable cryptography. It discusses only the quantum model, while our work is a comparative study of two no-cloning technologies with the goal of motivating a new direction that combines both of them. We view [22] as a complementary work as it contains more extensive details for the quantum setting.

2 Two Unclonability Paradigms

2.1 Quantum Computing

A quantum computer operates on quantum states that are represented as complex vectors. Computing over these states is captured by linear operators on appropriate complex spaces. The fundamental quantum unit of datum is the qubit, which is any vector in the unit sphere of $\mathbb{C}^2$. Such a vector can be written as a linear combination $\alpha|0\rangle + \beta|1\rangle$, where $|0\rangle$ and $|1\rangle$ are orthogonal unit vectors in $\mathbb{C}^2$ and α, β are complex numbers satisfying $||\alpha||^2 + ||\beta||^2 = 1$. Any state can be expressed as (complex) linear combinations of (tensor) products of individual qubits. Often a state is a linear combination of other states, or a combination of all states in a set or class of interest, a notion known as superposition which is a key feature of quantum computing. A valid quantum operation over quantum states is a linear transform over the appropriate spaces, and must map the input states into also well-defined states. Such valid maps are also known as unitaries, and capture one kind of operations typically used by quantum circuits.

Measurements. The other prominent class of quantum operations are *measurements*. These allow obtaining useful information from quantum states, and are tied to the notion of superposition. Measurements help test which particular representative state a given unknown state is most close to. This is akin to taking the projection of a given state over these representative states, and thus known as projective measurements. Even in the most general sense, measurements have the following crucial properties: they are destructive so that certain information about the state being measured is irrevocably lost (e.g., one cannot recover the starting superposition after a projective measurement, by definition), and hence, measurements are also irreversible. Unitaries, on the other hand, are reversible since the result of a unitary action can be undone by applying its adjoint.

Computing. Quantum circuits consist of a sequence of quantum unitaries and measurements (these are treated as gates in the circuit). If the data is carefully structured, this can imply that one can undo a quantum computation. That is, customarily, measurements are performed only at the end of a quantum circuit (otherwise, if performed earlier, uncomputation does not hold due to the destructive nature of measurements). Typically, a quantum circuit will use ancillary quantum registers in addition to the input register in order to operate.

Furthermore, some quantum algorithms involves running classical algorithms over quantum inputs; this is referred to as running such a program coherently or in superposition. Intuitively, this refers to interpreting basis states as bit strings, and interpreting the classical circuit as a quantum one that performs the desired classical operation on each basis state (and naturally on linear combinations).

No-cloning, rewinding, and more. All these facts imply several characteristic traits of quantum computing, such as the celebrated *no-cloning theorem*. It states that given an unknown state, there is no quantum circuit that can copy this state. Harnessing this principle is the basis for building unclonable cryptography in the quantum model as we discuss later. Another famous implication of this theorem is positing the infeasibility of *rewinding* a quantum algorithm as understood in the classical sense. Furthermore, quantum computing acts as a stage for advanced algorithms such as Grover search and Simon's algorithm, which have no classical counterparts, and lead to the notion of *quantum speedups*. The latter is the phenomenon in which one can find a quantum algorithm to solve a given computational task that runs much faster than any corresponding classical algorithm.

2.2 Unclonable Polymers

Advances in biotechnology have allowed the synthesis of biological polymers for the purpose of data storage. Most effort so far has focused on DNA; DNA evolved to be clonable, thus the stored message can be copied and read as many times as desired. The work in [4] explored another biological polymer, proteins, which similarly can be used to store digital data; the message is encoded into a sequence of amino acids that is synthesized to produce the protein material.

However, proteins possess unique features that are very promising from a cryptographic point of view. First, there is no way to clone a protein that is given in a small amount. Second, retrieving the digital data out of a protein is a destructive process. The standard lab procedure for this is mass spectrometry, which requires a macroscopic pure sample, i.e., one that contains the target protein with high purity. In order to determine the protein sequence, mass spectrometry will break the chemical bonds in the protein, thus destroying the sample (i.e., rewinding is not possible). Moreover, this machine either outputs the protein sequence (which is basically the encoding of the digital message) or nothing.

Biological consumable memory tokens. Almashaqbeh et al. utilize these features to build consumable memory tokens. They devise a protocol that stores a secret message as a protein material, such that retrieving this message requires a biological secret key. Storage begins with synthesising the secret message into protein. It then connects this protein with a shorter protein sequence called a header, such that this header can be recognized by the matching antibodies. The protein-header chain is then mixed with a large quantity of decoy proteins which are similarly structured but encode random keys and messages. The resulting sample is then put in a vial, and so this vial is the memory token.

Recovering the secret message from the vial is done using a chemical reaction in which the message protein is pulled down using the antibodies matching the attached header. Then, the header is cleaved, and the digital message can be retrieved by sequencing the target protein using mass spectrometery. Each data retrieval attempt will consume part of the vial due to the destructive nature of sequencing. After a few attempts, the whole token will be consumed, enforcing a *bounded data retrieval query* notion.

This protocol can be extended to produce tokens that store v secret messages. The token is engineered in a way that even a party who knows the secret keys (i.e., antibodies) of these messages, can retrieve only n secret messages where $n < v$.

Differences from idealized one-time memory devices. Due to limitations on the biology side, the protocol above builds tokens with weak parameters. They can only store a few messages, of short length, under short keys, with non-negligible soundness errors. That is, an adversary—who does not know the secret key—may try an incorrect one and obtain the message with some non-negligible probability. Moreover, the model of [4] accounts for powerful adversaries who may own more powerful machinery than what the provisioned honest party may have. This leads to a power gap; the honest party can perform one data retrieval query, while the adversary might be able to perform up to n queries. Thus, employing consumable tokens in sophisticated applications requires additional techniques to amplify their security guarantees.

2.3 A Comparison

In the quantum setting, unclonability is achieved through specific properties of carefully selected families of quantum states. Such states (i) can be verified—test that certain conditions are satisfied—using a specific circuit or oracle, and (ii) cannot be cloned by a quantum polynomial time adversary despite access to this circuit or oracle. A state of this type can be used as a verification token, or to encode certain data making this data unclonable, but it cannot by itself represent a program to obtain unclonable programs. On the self-destruction side, unclonable states usually are not destroyed through the process of verification, allowing for as many queries as desired. For an untampered state, this is a minimal disturbance leaving the state essentially unchanged (as in the gentle measurement lemma [1]). Only measurements that significantly disturb a state will cause destruction of information.

In the unclonable polymer setting, unclonability is only about data storage; polymers that carry data cannot be cloned. Until now there is no non-destructive measurement or superposition making the output of a data retrieval query vary based on the selected projections. Self-destruction always takes place whenever a message stored in a protein is retrieved, and either the original message is produced or nothing.

Table 1. Classification of unclonable primitives (q; quantum, $i\mathcal{O}$; indistinguishability obfuscation, OWF; one way function, LWE; learning with errors, CRS; common refernce string, and ROM; random oracle model).

Setting	**Paradigm**	**Existing Primitives**	**Additional Assumptions**
Quantum	Unclonable states	Quantum money	qOWF, $qi\mathcal{O}$, qLWE
	Unclonable programs	Copy protection	$qi\mathcal{O}$, qOWF, qLWE
	Unclonable programs	Secure software leasing	CRS, $qi\mathcal{O}$, qLWE
	Unclonable states	One-shot signatures	$qi\mathcal{O}$, any secure classic signature scheme
	Unclonable data	Unclonable encryption	qOWF, $qi\mathcal{O}$, qROM
	Unclonable programs	Unclonable decryption	$qi\mathcal{O}$, qOWF, qLWE
Polymers	Unclonable data	Digital lockers	ROM
	Unclonable programs	$(1, n)$-time programs	OWF, $i\mathcal{O}$

3 Building Unclonable Cryptography

Unclonable cryptography aims for building cryptographic primitives that enjoy some form of unclonability. Thus, at any point of time, only one party can invoke an algorithm when operating on an unclonable state or token. This strong concept resulted in several primitives, which we explore in this section and devise new constructions for them. In particular, for each primitive we explain its functionality and security guarantees, and explore existing constructions (quantum- or unclonable polymer-based)—which is done at a high level for a few representative contruction due to space limitation. We then introduce an alternative construction using the other unclonability paradigm (if possible). The goal is to delineate the relation between the two unclonability technologies.

Construction paradigms. As Table 1 shows, in the quantum setting, the underlying techniques of unclonable cryptography can be classified into three tiers, in increasing order of complexity and realized functionality (this organization was suggested by a discussion in [11]): *Tier one* involves *verifiable unclonable states*, with a (publicly accessed) circuit or oracle, as mentioned previously. Establishing the right kind of unclonability, when not shown directly, relies on (usually information theoretic) properties of the underlying states such as direct-product hardness. In *tier two—unclonable data*, techniques of data encoding compatible with the unclonable structures are used. In general, this paradigm often depends on arguably stronger underlying properties of the quantum states such as monogamy of entanglement. Moreover, the design of the schemes under this paradigm gets more elaborate: the quantum state is constructed based on the data to be encoded, while the associated data is encoded classically. Lastly, in *tier three—unclonable computation*—the previous states are associated with obfuscated circuits such that without these states, i.e., without the data they store or without a successful verification, the program will not execute.

In the unclonable polymer setting, the paradigms that apply are *tier two—unclonable data* and *tier three—unclonable programs*. A consumable token is used to store secret data, like a form of encryption, or unique random strings that can unlock an obfuscated program to execute over a given input. Naturally, realizing fully-fledged functionalities requires additional assumptions as the table shows.[4]

3.1 Bounded-execution Programs

One-time programs, or more generally k-time programs, are secret programs or these with some hardwired secrets that can be executed on up to k distinct inputs. They can be used in several applications, such as authentication, software protection, and obfuscating (learnable) functions that can be learned using a polynomial number of oracle queries.

Idealized k-time programs. This notion was put forward by Goldwasser et al. [17]. They show a construction that allows building a k-time program for any functionality by combining garbled circuits with one-time memory gadgets. At a high level, instead of engaging in an interactive oblivious transfer protocol so the recipient can obtain the garbled circuit labels of her input, the 0 and 1 labels of each wire are stored in a gadget. The recipient, in turn, can retrieve only one of these labels, after which the gadget stops responding. To support k-time executions, k garbled circuits are generated along with ℓk memory gadgets (for input length of ℓ bits).

Impossibility of k-time programs in the quantum model. Broadbent et al. [10] show that, similar to the classical model, k-time quantum programs are impossible without the one-time memory gadgets. This is because, without loss of generality, any measurement to compute the output can be done at the end, and by the gentle measurement lemma [1] and correctness of the scheme, the end measurement can be only mildly destructive. Thus, the evaluator can uncompute the evaluation to re-obtain the original program, and simply execute it over and over again.

$(1, n)$-time programs in the unclonable polymer model. This is a weaker version of one-time programs built using the real-world consumable tokens rather than the hypothetical one-time memory gadgets. Here, an honest recipient can execute the program over only one input, while an adversary can execute over n inputs. Such a power gap is inherited from the consumable token functionality. The construction relies on indistinguishability obfuscation ($i\mathcal{O}$) and pseudorandom number generators (PRGs); it associates a unique secret message m to each input x such that the obfuscated program must see the correct m in order to execute over x. Beside obfuscating the program, the sender creates a token containing all the messages corresponding to the input space of f. Thus, for an input x, the recipient uses the corresponding token key to retrieve the message m. An attacker

[4] Note that the table lists *all* additional assumptions used by the constructions we review for each primitive (just to give a sense of complexity/difficulty). A particular construction may not need all these assumptions.

who might seize the program and the token, will be only able to retrieve up to n messages from the token (corresponding to n distinct inputs), and execute f over these inputs. This construction is extended with linear error correcting codes to allow handling functions with exponential size input space.[5]

3.2 Quantum Money

Quantum money [24] is a (theoretical) digital currency that prevents double spending by definition. The basic idea is to represent a money token as an unclonable state tied to a unique identifier such as a serial number. Typically, a bank is involved in issuing and managing currency. Quantum money schemes can be private (i.e., tokens can be verified by the bank only) or public (can be verified by anyone).

Existing constructions in the quantum model. In the original [24] construction, the money state is created using a quantum-secure PRF evaluated on a given (unique) serial number. The bank holds the secret minting key, which is the key for the PRF. Another private scheme [19] uses pseudorandom states (PRSs) as the money states, which are produced using a PRS key (similar to a PRG seed, except that the output of a PRS is a quantum state), and verification is done using a reflection oracle for the produced state. For publicly verifiable quantum money, [2] show an oracle-aided construction based on subspace states. They showed that these states are unique in passing a membership test over the primal and dual subspaces.[6] So, given only black-box access to such membership oracles—where these oracles can be constructed using quantum-secure $i\mathcal{O}$ [25]—unforgeability of money is infeasible. Recently, Shmueli [23] employed quantum FHE, $qi\mathcal{O}$, and subexponential quantum hardness of learning with error (LWE) to build a public quantum money construction that supports classical certificates of destruction.

The unclonable polymer setting. Quantum money has the basic feature of one-time spending. Unfortunately, consumable tokens cannot be used here due to the power gap between the adversary and the honest party. Regardless of how consumable tokens are combined with other assumptions, an honest, indeed, will spend a coin once, but an attacker can perform that operation up to n times.

3.3 Software Copy Protection

In this primitive, a given program f is encoded into a state $|\psi_f\rangle$ that allows evaluating f on any desired input, but does not permit cloning $|\psi_f\rangle$. Not all functions admit this notion, for example, learnable functions cannot be copy protected.

[5] Recall that a consumable token has a non-negligible soundness error γ. This error must be first amplified to negligible, which can be done using secret sharing.

[6] Given a subspace $A \in \mathbb{F}_2^n$, its state $|A\rangle$ is the uniform superposition of all strings in A (with the application of a Hadamard in all qubits creates the orthogonal complement subspace state $|A^\perp\rangle$). Unclonability of $|A\rangle$ holds even when given oracle access to the membership in A and $A^\perp$.

Existing constructions in the quantum model. The notion of quantum copy protection was introduced in [1] with two candidate schemes for any unlearnable functionality. Basically, a specific *quantum oracle* that is created based on the function being copy protected. The work [3] gives another scheme with respect to a (pair of) *classical* oracle(s): a random oracle and another that computes the function output masked by the output of the random oracle. The actual copy-protected program consists of a subspace state, and the oracles run verification tests before outputting the function value. Given the subspace state, the evaluator can run these oracles (coherently) in turn on this state and obtain the outputs of both oracles.

Without oracles, it is impossible to have copy-protection for the class of all unlearnable functions in the plain model; a result that was proved by Ananth et al. [8]. This led to targeting particular function classes instead. For example, Coladangel et al. [15] show a copy-protection scheme for decryption circuits, PRFs and signing circuits using (properties of) hidden coset states and $qi\mathcal{O}$, as well as quantum secure one-way functions and quantum hardness of LWE. The copy-protected function includes an obfuscated circuit that tests for the well-formedness of the coset state, and then computes the desired function using hardwired secrets in the circuit.

The unclonable polymer setting. Consumable tokens alone cannot support this application, but instead $(1, n)$-time programs should be used. That is, the (classical) software to be protected will be compiled into a $(1, n)$-time program. This leads to limitations: first, this protected software is a bounded execution one, while copy protection allows the recipient to execute as many times as she wishes. Second, the adversary can perform n executions of the program, while the honest recipient can execute only once. Third, an adversary can create several copies of the $(1, n)$-time program (simply copy the obfuscated program), and perform a splitting attack in which the consumable token material is divided among several parties. This may lead into splitting the domain of the encapsulated function f into subsets $T_1, \ldots, T_u$, and each T_i can be given to a separate attacker. Although this is not a full copy of the program, now we have more than one adversary able to execute the program simultaneously (albeit on disjoint sets of inputs).

3.4 Secure Software Leasing

The work of [8] shows a related but weaker unclonable security notion called secure software leasing (SSL). Here, the adversary can clone the protected program, but may not create new copies that can be authenticated with respect to a certain procedure specified alongside the scheme. Two flavors of this notion were formulated: finite and infinite term lessor security. Finite term specifies that the copies are leased out for a specified amount of time and must be returned, and the lessee cannot then produce an authentic copy of the program after that. While in the infinite term, the adversary does not return the leased state but still cannot produce any authentic copies of the program apart from her own. [8] show that SSL for general functionalities, similar to copy protection, is impossible in the plain model.

Existing constructions in the quantum model. [8] provide a construction for an infinite-term SSL scheme for searchable compute-and-compare circuits (which can be thought of as augmented point functions, and include the class of all point functions and all conjunctions with wildcards), assuming subspace obfuscators and the subexponential quantum hardness of LWE. This construction is in the CRS model, and uses simulation-extractable none-interactive zero knowledge proofs (NIZKs) and input-hiding obfuscators (which hide the key input for evasive functions) along with subspace states. [20] build on this approach to obtain an SSL construction for PRFs from weaker assumptions, namely just quantum hardness of LWE. Broadbent et al. [11] build an unconditional SSL construction (i.e., without any assumptions) again for the same class as [8], satisfying a slightly weaker and distributional finite term security guarantee.

The unclonable polymer setting. Same argument as software copy protection above. However, for finite term security, there is no need to return a program, the token will be consumed after being used.

3.5 One-shot Signatures

One-shot signatures [5] allow creating only one valid signature under a given public key, or alternatively, a signing key can be used to produce only one signature. For delegation, this is envisioned as signature tokenization; a delegator generates a signing token that a delegatee transforms into a signature over a message m of her choice. The one-time is achieved by generating an unclonable token; once it is used to sign m, it self-destructs.

Existing constructions in the quantum model. The construction in [9] relies on quantum subspace states to create one-time signing tokens. Alice has the secret key the subspace A, and gives Bob the state $|A\rangle$ as the signing token. To sign message 0, Bob outputs a string $v \in A$ by measuring $|A\rangle$ in the computational basis, and to sign message 1 Bob outputs a string $v' \in A^{\perp}$ by measuring $|A\rangle$ in the Hadamard basis. Since this measurement consumes the state, Bob can sign only once. This construction has been improved in [15] to rely on coset states and $i\mathcal{O}$ to instantiate the oracle.

Amos et al. [5] showed another construction using the hash-then-sign paradigm based on a (quantum) Chameleon hash function—a hash function that allows finding collisions given a secret trapdoor. Basically, Bob samples a hash value h with a quantum secret key (or trapdoor) $|\mathsf{sk}\rangle$. Alice signs h and sends the (classical) signature σ back to Bob. To sign m, Bob can use $|\mathsf{sk}\rangle$ to equivocate h to become a hash of m (which is basically computing r such that $H(m, r) = h$), and output (σ, r) as the signature). This equivocation can be called only once since $|\mathsf{sk}\rangle$ will be destructed after the first call.

The unclonable polymer setting. A potential construction is the following. If the set of messages from which m will be chosen is known, then the Chameleon hash-based idea can be used. Alice generates all the equivocation randomness for these messages and store them in a consumable token, then Bob will retrieve the random string that corresponds to the message he wants to sign. Of course, an

attacker can retrieve n random strings, thus sign n messages instead of one. Any other construction will suffer from a similar drawback, which is inherent due to the power gap of consumable tokens. For this reason, consumable tokens (and $(1, n)$-time programs) cannot be used to construct one-shot signatures.

3.6 Unclonable Encryption

Unclonable encryption has classical encryption/decryption keys but produces unclonable ciphertexts so that only one party at a time can decrypt. The security guarantee is that an adversary cannot manipulate a ciphertext such that two (non-interacting) parties, even if given the decryption key, can correctly decrypt.

Existing constructions in the quantum model. In [12], two symmetric unclonable encryption constructions were proposed. The first generates a secret key composed of two random strings (r, θ) for each message to be encrypted. The encryption is a quantum state $\rho = |(m \oplus r)^\theta\rangle\langle(m \oplus r)^\theta|$. To decrypt, compute $\rho' = H^\theta \rho H^\theta$, then measure ρ' to obtain c and compute $m = c \oplus r$. The second (reusable key) scheme works in the random oracle model and uses a quantum-secure PRF f. Here, the encryption key is composed of two random strings (s, θ). For each message m, choose a fresh random string r and set the ciphertext as $\rho = |r^\theta\rangle\langle r^\theta|$ and the classical string $c = m \oplus f_s(r)$. To decrypt, measure the state ρ to produce r, and compute $m = c \oplus f_s(r)$. This measurement will destroy the state, and an attacker cannot split this state to have two ciphertexts that can be correctly decrypted. Followup works avoided the need for a random oracle [6] or kept that but introduced stronger security notions [7].

The unclonable polymer setting. Symmetric unclonable encryption can be built in a straightforward way using a consumable token. The decryption key is mapped to a token key, and the secret message m is stored in the token under this key. In order to decrypt, the recipient simply performs a data retrieval query. However, an attacker can perform a splitting attack; split each token into n samples, each of which is a valid ciphertext. Thus, and following the definition of the encryption unclonability game, n non-interacting attackers can decrypt once the decryption key is revealed. As a result, the unclonable polymer setting realizes a modified security notion that can be called n-unclonable encryption, where given a (polymer) ciphertext, it cannot produce $n + 1$ (rather than just 2) ciphertexts each of which can correctly decrypt to m.

3.7 Unclonable Decryption

Unclonable decryption, or single-decryptor encryption [16], has a classical encryption key and a quantum decryption $|\mathsf{sk}\rangle$. Given this secret key only one party can decrypt a given ciphertext at a time.

Existing constructions in the quantum model. [16] introduced two constructions. The first is a symmetric one, which is basically the construction of unclonable encryption from [12], but with a classical ciphertext and a quantum state decryption key. The second is a public key encryption scheme that relies

on one-shot signatures and extractable witness encryption. To encrypt m, pick a randomness r and witness encrypt m with respect to the NP statement (r, σ) (where σ is a valid signature over r). To decrypt, sign r then decrypt m using the witness (r, σ). Re-usability of the signing key $|\mathsf{sk}\rangle$ is achieved by computing the signing and decryption on the superposition and measuring only the quantum state of the decrypted m. This allows uncomputing to retrieve $|\mathsf{sk}\rangle$ and use it for another decryption. [15] also include a construction for unclonable decryption based on their copy protection approach, that makes the relatively milder assumptions of $qi\mathcal{O}$ and $q\mathsf{OWF}$.

The unclonable polymer setting. The same observation used in [16] can be applied here. That is, send a classical ciphertext and store the decryption key in a consumable token instead. Due to the splitting attack, a similar modified security notion applies here as in unclonable encryption. That is, challenge the adversary to produce more than n decryptors (rather than more than one decryptor as in the original notion).

3.8 Bounded-query Digital Lockers

In a digital locker [13] a secret message m is encrypted using a low-entropy password such that the only feasible attack is an exhaustive search over the password space. This is a point function with multibit output: for password space $\mathcal{P}$ and message space $\mathcal{M}$, $I_{p,m} : \mathcal{P} \to \mathcal{M} \cup \{\perp\}$ is a point function that outputs m when queried on p and $\perp$ otherwise. The question is whether we can build a digital locker that resists exhaustive search attacks.

Existing construction in the unclonable polymer setting. With consumable tokens, [4] showed how to build digital lockers that limit the number of password guesses an attacker can make. At a high level, the password p is mapped to a token key at random, and then the message m is stored under this key inside a token. The mapping function is public, and so the recipient can use the password to retrieve m. An attacker will be able to perform up to n data retrieval queries, and so try up to n password guesses but not more. To amplify the soundness error of the token to negligible, and so limit the success probability to be almost $n/|\mathcal{P}|$, [4] use secret sharing and a chaining technique to force an adversary to operate on the tokens (holding the shares of m) sequentially, and so preserve the number of password guesses to be n.

The quantum model. Digital lockers cannot be supported in the quantum model due to the impossibility of bounded execution without one-time memory gadgets, which applies here. Namely, an attacker can evaluate over her password guess, uncompute appropriately, then try a new guess, and basically repeat this process as many times as desired.

4 The Road Ahead

Incomparable technologies. As observed, our study indicates that these two unclonable technologies are somewhat incomparable and often involve orthogonal

Table 2. Unclonable primitives and bounded execution programs.

Primitive to realize	**Using k-time programs?**	**Using $(1, n)$-time programs?**
Quantum money	Yes (with $k = 1$)	No—a coin can be spent n times
Software copy protection (and secure software leasing)	Yes (including learnable functions)—but a program can be executed only k times	Yes—but permitting domain splitting attacks and the power gap between the honest party and the adversary
One-shot signatures	Yes (with $k = 1$)	No—an attacker can sign up to n messages instead of one
Unclonable encryption	Yes	Yes—but a weaker security notion covering $n + 1$ attackers instead of two
Unclonable decryption	Yes	Yes—same constraint as above
Digital lockers	Yes—k trials for honest party	Yes

principles. Bounded execution programs can be realized in the unclonable-polymer setting, but they are impossible in the quantum model without trusting tamper-proof hardware. At the same time, the power gap between the honest party and the adversary makes it impossible to realize particular primitives, such as one-shot signatures, or offers weaker security notions, such as the n-unclonability for unclonable decryption. On the other hand, quantum-based constructions of primitives, such as copy protection and secure software leasing, have the desirable property of *persistence*, i.e., the protected program is reusable as it does not degrade necessarily after use.

The power of bounded execution programs. Bounded execution programs are unclonable by definition. Their generalized nature, where they can encapsulate an arbitrary functionality, raises the question of whether they can realize existing unclonable cryptographic primitives. As Table 2 shows, the answer is usually *yes* for idealized k-time programs, but at the expense of losing the persistence feature when desired. On the other hand, for $(1, n)$-programs, the inherent power gap makes the answer either *no* or *conditional yes* due to capability trade-offs.

A stronger unclonable polymer model? A natural question that comes to mind is around strengthening the unclonable polymer setting. That is, is there a way to close the power gap so both an honest and an adversary obtain the same number of data retrieval queries? On the one hand, this will lead to idealized k-time programs, as well as one-shot unclonable primitives using their conventional security notions, in the polymer model. On the second hand, a consumable token without this power gap implements non-interactive oblivious

transfer,[7] which is impossible in the standard model. We believe that such a huge advancement would require more on the biology side.

A hybrid model. Combining both technologies appears to potentially be a very fruitful direction realizing both copy protection and bounded query. A starting point could be the use of consumable tokens, $qi\mathcal{O}$, and qOWF, in an analogues way to [4], to build $(1, n)$-time quantum programs. Another is exploring the new notions of n-unclonable encryption/decryption in the polymer setting, and investigating their applications in building new flavors of secure function evaluation and non-interactive cryptographic protocols.

This hybrid approach, of course, requires new storage and communication models, and new security notions accounting for the differences between the two technologies in terms of adversary classes/capabilities and hardness assumptions. We leave such considerations to future work, and limit ourselves to the observation that there seem to be no natural or fundamental barriers to such an approach.

On the nature of emergent technologies: We exercise a cryptographer's skepticism to add a note of caution; none of what we have covered here should be seen as definitive in terms of eventual real-world applications. There are *considerable* challenges and unseen, yet-to-appear problems in the course of realizing either viable quantum computers, or protein data tokens. Each involves physical, societal and ethical conundrums, and the end product in either case may take a very different form to what we currently suppose. For example, the bounded query feature obtained from proteins is a consequence of both their unclonability and destructive sequencing. If new advances lead to non-destructive sequencing machinery, then this feature will not hold anymore. At the same time, looking for and harnessing emergent technologies and unexpected developments also give rise to unexpected cryptography in their own right!

References

1. Aaronson, S.: Quantum copy-protection and quantum money. In: CCC (2009)
2. Aaronson, S., Christiano, P.: Quantum money from hidden subspaces. In: STOC (2012)
3. Aaronson, S., Liu, J., Liu, Q., Zhandry, M., Zhang, R.: New approaches for quantum copy-protection. In: CRYPTO (2021)
4. Almashaqbeh, G., Canetti, R., Erlich, Y., Gershoni, J., Malkin, T., Pe'er, I., Roitburd-Berman, A., Tromer, E.: Unclonable polymers and their cryptographic applications. In: EUROCRYPT (2022)
5. Amos, R., Georgiou, M., Kiayias, A., Zhandry, M.: One-shot signatures and applications to hybrid quantum/classical authentication. In: STOC (2020)
6. Ananth, P., Kaleoglu, F.: Unclonable encryption, revisited. In: TCC (2021)

[7] In its simplest form, 1-out-of-2 OT is protocol involving a sender with two secret message m_0 and m_1, and a recipient with input bit b; the receiver will only retrieve m_b and the sender will not know which message was retrieved.

7. Ananth, P., Kaleoglu, F., Li, X., Liu, Q., Zhandry, M.: On the feasibility of unclonable encryption, and more. In: CRYPTO (2022)
8. Ananth, P., La Placa, R.L.: Secure software leasing. In: EUROCRYPT (2021)
9. Ben-David, S., Sattath, O.: Quantum tokens for digital signatures. arXiv preprint arXiv:1609.09047 (2016)
10. Broadbent, A., Gutoski, G., Stebila, D.: Quantum one-time programs. In: CRYPTO (2013)
11. Broadbent, A., Jeffery, S., Lord, S., Podder, S., Sundaram, A.: Secure software leasing without assumptions. In: TCC (2021)
12. Broadbent, A., Lord, S.: Uncloneable quantum encryption via oracles. In: TQC (2020)
13. Canetti, R., Dakdouk, R.R.: Obfuscating point functions with multibit output. In: EUROCRYPT (2008)
14. Chen, Z., Vasilakis, G., Murdock, K., Dean, E., Oswald, D., Garcia, F.D.: {VoltPillager}: Hardware-based fault injection attacks against intel {SGX} enclaves using the {SVID} voltage scaling interface. In: USENIX Security (2021)
15. Coladangelo, A., Liu, J., Liu, Q., Zhandry, M.: Hidden cosets and applications to unclonable cryptography. In: CRYPTO (2021)
16. Georgiou, M., Zhandry, M.: Unclonable decryption keys. Cryptology ePrint Archive (2020)
17. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-time programs. In: CRYPTO (2008)
18. Gottesman, D.: Uncloneable encryption. arXiv preprint quant-ph/0210062 (2002)
19. Ji, Z., Liu, Y.K., Song, F.: Pseudorandom quantum states. In: CRYPTO (2018)
20. Kitagawa, F., Nishimaki, R., Yamakawa, T.: Secure software leasing from standard assumptions. In: TCC (2021)
21. Pappu, R., Recht, B., Taylor, J., Gershenfeld, N.: Physical one-way functions. Science **297**(5589), 2026–2030 (2002)
22. Sattath, O., Wyborski, S.: Uncloneable decryption from quantum copy protection. arXiv preprint arXiv:2203.05866 (2022)
23. Shmueli, O.: Public-key quantum money with a classical bank. In: STOC (2022)
24. Wiesner, S.: Conjugate coding. SIGACT News **15**(1), 78–88 (1983)
25. Zhandry, M.: Quantum lightning never strikes the same state twice. In: EUROCRYPT (2019)