A Modular Approach to Unclonable Cryptography

Prabhanjan Ananth*
UCSB

Amit Behera†
Ben-Gurion University

Abstract

We explore a new pathway to designing unclonable cryptographic primitives. We propose a new notion called unclonable puncturable obfuscation (UPO) and study its implications for unclonable cryptography. Using UPO, we present modular (and arguably, simple) constructions of many primitives in unclonable cryptography, including, public-key quantum money, quantum copy-protection for many classes of functionalities, unclonable encryption and single-decryption encryption.

Notably, we obtain the following new results assuming the existence of UPO:

- We show that any cryptographic functionality can be copy-protected as long as this functionality satisfies a notion of security, which we term as puncturable security. Prior feasibility results focused on copy-protecting specific cryptographic functionalities.
- We show that copy-protection exists for any class of evasive functions as long as the associated distribution satisfies a preimage-sampleability condition. Prior works demonstrated copy-protection for point functions, which follows as a special case of our result.
- We show that unclonable encryption exists in the plain model. Prior works demonstrated feasibility results in the quantum random oracle model.

We put forward a candidate construction of UPO and prove two notions of security, each based on the existence of (post-quantum) sub-exponentially secure indistinguishability obfuscation and one-way functions, the quantum hardness of learning with errors, and a new conjecture called simultaneous inner product conjecture.

*prabhanjan@cs.ucsb.edu
†behera@post.bgu.ac.il

Contents

1 Introduction

Unclonable cryptography leverages the no-cloning principle of quantum mechanics [WZ82, Die82] to build many novel cryptographic notions that are otherwise impossible to achieve classically. This has been an active area of interest since the 1980s [Wie83]. In the past few years, researchers have investigated a dizzying variety of unclonable primitives such as quantum money [AC12, Zha19, Shm22, LMZ23] and its variants [RS19, BS20, RZ21], quantum one-time programs [BGS13], copy-protection [Aar09, CLLZ21], tokenized signatures [BS16, CLLZ21], unclonable encryption [Got02, BL20] and its variants [KN23], secure software leasing [AL21], single-decryptor encryption [GZ20, CLLZ21], and many more [BKL23, GMR23, JK23].

However, establishing the feasibility of unclonable primitives has been quite challenging. The adversarial structure considered in the unclonability setting (i.e., spatially separated and entangled) is quite different from what we typically encounter in the traditional cryptographic setting. This makes it difficult to leverage traditional classical techniques, commonly used in cryptographic proofs, to argue the security of unclonable primitives. In some unclonable primitives (for example, unclonable encryption [BL20]) the secret key is revealed at a later stage to the adversary which presents another challenge in the usage of cryptographic tools to build these unclonable primitives. As a consequence, proofs in the area of unclonable cryptography tend to be complex and use sophisticated tools, making the literature less accessible to the broader research community. Moreover, there are still many important open problems in the area that are yet to be resolved. One such open problem (mentioned in [BL20, AKL+22]) is the feasibility of unclonable encryption in the plain model. Another open problem (mentioned in [ALL+21]) is identifying the classes of functionalities that can be copy-protected.

Overarching goal of our work. We advocate for a modular approach to designing unclonable cryptography. Our goal is to identify a central cryptographic primitive that leads to the simplified design of many unclonable primitives. Ideally, we would like to abstract away all the complex details in the instantiation of this central primitive and it should be relatively easy, even to classical cryptographers, to use this primitive to study unclonability in the context of other cryptographic primitives. Our hope is that the identification and instantiation of such a central primitive would speed up the progress in the design of unclonable primitives.

There is merit to exploring the possibility of such a central primitive in unclonable cryptography. Similar explorations in other contexts, such as classical cryptography, have been fruitful. For instance, the discovery of indistinguishability obfuscation [BGI+01, GGH+16] revolutionized cryptography and led to the resolution of many open problems (for instance: [SW14, GGHR14, BZ17, BPR15]).

Summary of our results. Towards implementing a modular approach, we propose a new notion of program obfuscation called *unclonable puncturable obfuscation* (UPO). Using UPO, we present the constructions of many primitives in unclonable cryptography. En route, we discover new feasibility results related to copy-protection and unclonable encryption. We propose two constructions of UPO and base their security on well-studied cryptographic assumptions and novel conjectures.

1.1 Our Results

Definition. We discuss our results in more detail. Roughly speaking, unclonable puncturable obfuscation (UPO) defined for a class of circuits $\mathfrak{C}$ in $\mathsf{P/Poly}$, consists of two QPT algorithms $(\mathsf{Obf}, \mathsf{Eval})$ defined as follows:

- Obfuscation algorithm: Obf takes as input a classical circuit $C \in \mathfrak{C}$ and outputs a quantum state ρ_C.
- Evaluation algorithm: Eval takes as input a quantum state ρ_C, an input x, and outputs a value y.

In terms of correctness, we require $y = C(x)$. To define security, as is typically the case for unclonable primitives, we consider non-local adversaries of the form $(\mathcal{A}, \mathcal{B}, \mathcal{C})$. The security experiment, parameterized by a distribution $\mathcal{D}_{\mathcal{X}}$, is defined as follows:

- $\mathcal{A}$ (Alice) receives as input a quantum state ρ^* that is generated as follows. $\mathcal{A}$ sends a circuit C to the challenger, who the samples a bit b uniformly at random and samples $\left(x^{\mathcal{B}}, x^{\mathcal{C}}\right)$ from $\mathcal{D}_{\mathcal{X}}$. If $b = 0$, it sets ρ^* to be the output of Obf on input C, or if $b = 1$, it sets ρ^* to be the output of Obf on G, where G is a punctured circuit that has the same functionality as C on all the points except $x^{\mathcal{B}}$ and $x^{\mathcal{C}}$. It is important to note that $\mathcal{A}$ only receives ρ^* and in particular, $x^{\mathcal{B}}$ and $x^{\mathcal{C}}$ are hidden from $\mathcal{A}$.
- $\mathcal{A}$ then creates a bipartite state and shares one part with $\mathcal{B}$ (Bob) and the other part with $\mathcal{C}$ (Charlie).
- $\mathcal{B}$ and $\mathcal{C}$ cannot communicate with each other. In the challenge phase, $\mathcal{B}$ receives $x^{\mathcal{B}}$ and $\mathcal{C}$ receives $x^{\mathcal{C}}$. Then, they each output bits $b_{\mathcal{B}}$ and $b_{\mathcal{C}}$.

$(\mathcal{A}, \mathcal{B}, \mathcal{C})$ win if $b_{\mathcal{B}} = b_{\mathcal{C}} = b$. The scheme is secure if they can only win with probability at most 0.5 (ignoring negligible additive factors).

Keyed Circuits. Towards formalizing the notion of puncturing circuits in a way that will be useful for applications, we consider keyed circuit classes in the above definition. Every circuit in a keyed circuit class is of the form $C_k(\cdot)$ for some key k. Any circuit class can be implemented as a keyed circuit class using universal circuits and thus, by considering keyed circuits, we are not compromising on the generality of the above definition.

Challenge Distributions. We could consider different settings of $\mathcal{D}_{\mathcal{X}}$. In this work, we focus on two settings. In the first setting (referred to as *independent* challenge distribution), sampling $(x^{\mathcal{B}}, x^{\mathcal{C}})$ from $\mathcal{D}_{\mathcal{X}}$ is the same as sampling $x^{\mathcal{B}}$ and $x^{\mathcal{C}}$ uniformly at random (from the input space of C). In the second setting (referred to as *identical* challenge distribution), sampling $(x^{\mathcal{B}}, x^{\mathcal{C}})$ from $\mathcal{D}_{\mathcal{X}}$ is the same as sampling x uniformly at random and setting $x = x^{\mathcal{B}} = x^{\mathcal{C}}$.

Generalized UPO. In the above security experiment, we did not quite specify the behavior of the punctured circuit on the points $x^{\mathcal{B}}$ and $x^{\mathcal{C}}$. There are two ways to formalize and this results in two different definitions; we consider both of them in Section 3. In the first (basic) version, the output of the punctured circuit G on the punctured points is set to be $\perp$. This version would be the regular UPO definition. In the second (generalized) version, we allow $\mathcal{A}$ to control the output of the punctured circuit on inputs $x^{\mathcal{B}}$ and $x^{\mathcal{C}}$. For instance, $\mathcal{A}$ can choose and send the circuits $\mu_{\mathcal{B}}$ and $\mu_{\mathcal{C}}$ to the challenger. On input $x^{\mathcal{B}}$ (resp., $x^{\mathcal{C}}$), the challenger programs the punctured circuit G to output $\mu_{\mathcal{B}}(x^{\mathcal{B}})$ (resp., $\mu_{\mathcal{C}}(x^{\mathcal{C}})$). We refer to this version as *generalized UPO*.

Applications. We demonstrate several applications of UPO to unclonable cryptography.

We summarise the applications[1] in Figure 1. For a broader context of these results, we refer the reader to Appendix B (Related Work).

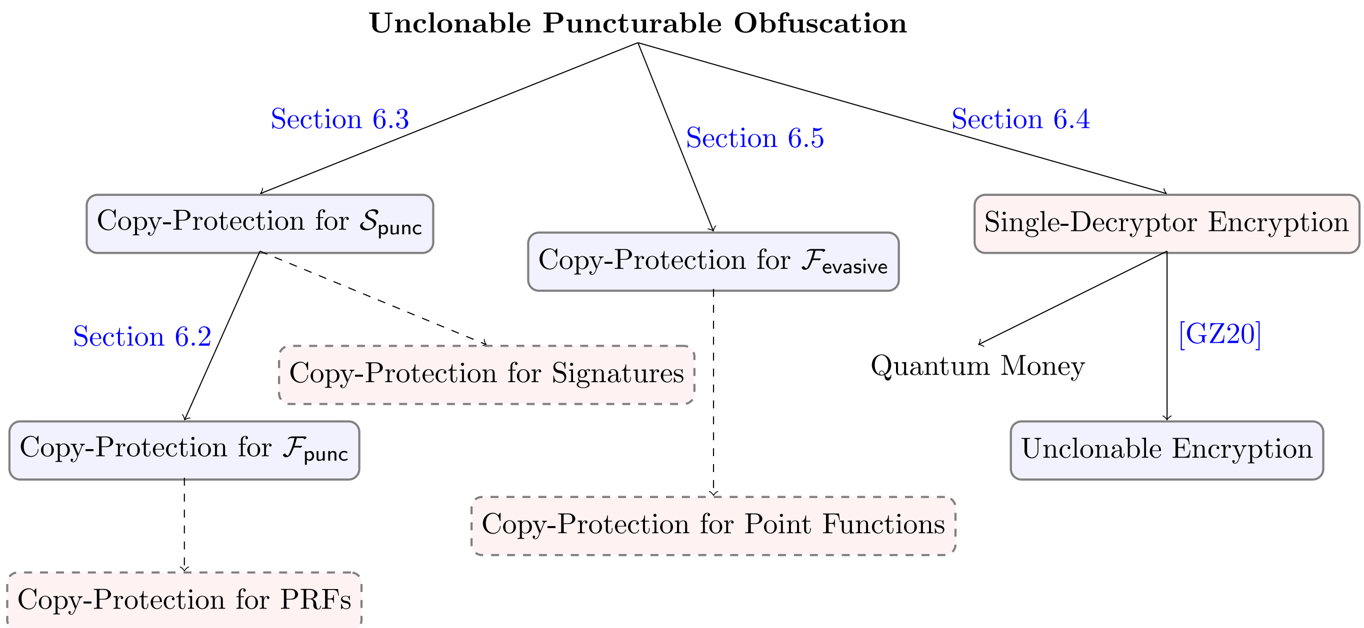

Figure 1: Applications of Unclonable Puncturable Obfuscation. $\mathcal{S}_{\mathsf{punc}}$ denotes cryptographic schemes satisfying puncturable property. $\mathcal{F}_{\mathsf{punc}}$ denotes cryptographic functionalities satisfying functionalities satisfying puncturable property. $\mathcal{F}_{\mathsf{evasive}}$ denotes functionalities that are evasive with respect to a distribution $\mathcal{D}$ satisfying preimage-sampleability property. The dashed lines denote corollaries of our main results. The blue filled boxes represent primitives whose feasibility was unknown prior to our work. The red filled boxes represent primitives for which we get qualitatively different results when compared to previous works.

Copy-Protection for Puncturable Cryptographic Schemes (Section 6.2 and Section 6.3). We consider cryptographic schemes satisfying a property called puncturable security. Informally speaking, puncturable security says the following: given a secret key sk, generated using the scheme, it is possible to puncture the key at a couple of points $x^{\mathcal{B}}$ and $x^{\mathcal{C}}$ such that it is computationally infeasible to use the punctured secret key on $x^{\mathcal{B}}$ and $x^{\mathcal{C}}$. We formally define this in Section 6.3.

We show the following:

[1] We refer the reader unfamiliar with copy-protection, single-decryptor encryption, or unclonable encryption to the introduction section of [AKL23] for an informal explanation of these primitives.

Theorem 1. *Assuming UPO for* P/poly*, there exists copy-protection for puncturable cryptographic schemes.*

Prior works [CLLZ21, LLQZ22] aimed at copy-protecting specific cryptographic functionalities whereas we, for the first time, characterize a broad class of cryptographic functionalities that can be copy-protected.

As a corollary, we obtain the following results assuming UPO.

- We show that **any** class of puncturable pseudorandom functions that can be punctured at two points [BW13, BGI14] can be copy-protected. The feasibility result of copy-protecting pseudorandom functions was first established in [CLLZ21]. A point to note here is that in [CLLZ21], given a class of puncturable pseudorandom functions, they transform this into a different class of pseudorandom functions[2] that is still puncturable and then copy-protect the resulting class. On the other hand, we show that *any* class of puncturable pseudorandom functions, which allows for the puncturing of two points, can be copy-protected. Hence, our result is qualitatively different than [CLLZ21].

- We show that **any** digital signature scheme, where the signing key can be punctured at two points, can be copy-protected. Roughly speaking, a digital signature scheme is puncturable if the signing key can be punctured on two messages $m^{\mathcal{B}}$ and $m^{\mathcal{C}}$ such that given the punctured signing key, it is computationally infeasible to produce a signature on one of the punctured messages. Our result rederives and generalizes a recent result by [LLQZ22] who showed how to copy-protect the digital signature scheme of [SW14].

In the technical sections, we first present a simpler result where we copy-protect puncturable functionalities (Section 6.2) and we then extend this result to achieve copy-protection for puncturable cryptographic schemes (Section 6.3).

COPY-PROTECTION FOR EVASIVE FUNCTIONS (SECTION 6.5). We consider a class of evasive functions associated with a distribution $\mathcal{D}$ satisfying a property referred to as preimage-sampleability which is informally defined as follows: there exists a distribution $\mathcal{D}'$ such that sampling an evasive function from $\mathcal{D}$ along with an accepting point (i.e., the output of the function on this point is 1) is computationally indistinguishable from sampling a function from $\mathcal{D}'$ and then modifying this function by injecting a uniformly random point as the accepting point. We show the following.

Theorem 2. *Assuming generalized UPO for* P/poly*, there exists copy-protection for a class of functions that is evasive with respect to a distribution $\mathcal{D}$ satisfying preimage-sampleability property.*

Unlike Theorem 1, we assume generalized UPO in the above theorem.

As a special case, we obtain copy-protection for point functions. A recent work [CHV23] presented construction of copy-protection for point functions from post-quantum iO and other standard assumptions. Qualitatively, our results are different in the following ways:

- The challenge distribution considered in the security definition of [CHV23] is arguably not a natural one: with probability $\frac{1}{3}$, $\mathcal{B}$ and $\mathcal{C}$ get as input the actual point, with probability $\frac{1}{3}$, $\mathcal{B}$ gets the actual point while $\mathcal{C}$ gets a random value and finally, with probability $\frac{1}{3}$, $\mathcal{B}$

[2] Spefically, they add a transformation to generically make the pseudorandom function extractable.

gets a random value while $\mathcal{C}$ gets the actual point. On the other hand, we consider identical challenge distribution; that is, $\mathcal{B}$ and $\mathcal{C}$ both receive the actual point with probability $\frac{1}{2}$ or they both receive a value picked uniformly at random.

- While the result of [CHV23] is restricted to point functions, we show how to copy-protect functions where the number of accepting points is a fixed polynomial.

We clarify that none of the above results on copy-protection contradicts the impossibility result by [AL21] who present a conditional result ruling out the possibility of copy-protecting contrived functionalities.

Unclonable Encryption (Section 6.4). Finally, we show, for the first time, an approach to construct unclonable encryption in the plain model. We obtain this construction by first constructing public-key single-decryptor encryption (SDE) with an identical challenge distribution. [GZ20] showed that SDE with such a challenge distribution implies unclonable encryption.

Theorem 3. *Assuming generalized UPO for* $\mathsf{P/poly}$*, post-quantum indistinguishability obfuscation (iO), and post-quantum one-way functions, there exists a public-key single-decryptor encryption scheme with security against identical challenge distribution.*

Prior work by [CLLZ21] demonstrated the construction of public-key single-decryptor encryption with security against independent challenge distribution, which is not known to imply unclonable encryption.

Apart from unclonable encryption, single-decryptor encryption also implies public-key quantum money. We thus, obtain the following corollaries.

Corollary 4. *Assuming generalized UPO, post-quantum iO, and post-quantum one-way functions, there exists a one-time unclonable encryption scheme in the plain model.*

We note that this is the first construction of unclonable encryption in the plain model. All the previous works [BL20, AKL$^+$22, AKL23] construct unclonable encryption in the quantum random oracle model. The disadvantage of our construction is that they leverage computational assumptions whereas the previous works [BL20, AKL$^+$22, AKL23] are information-theoretically secure.

Using the compiler of [AK21], we can generically transform a one-time unclonable encryption into a public-key unclonable encryption in the plain model under the same assumptions as above.

Corollary 5. *Assuming generalized UPO, post-quantum iO, and post-quantum one-way functions, there exists a public-key quantum money scheme.*

The construction of quantum money from UPO offers a conceptually different approach to construct public-key quantum money in comparison with other quantum money schemes such as [Zha19, LMZ23, Zha23].

As an aside, we also present a lifting theorem that lifts a selectively secure single-decryptor encryption into an adaptively secure construction, assuming the existence of post-quantum iO. Such a lifting theorem was not known prior to our work.

Construction. Finally we demonstrate a construction of generalized UPO for all classes of efficiently computable keyed circuits. We show that the same construction is secure with respect to both identical and independent challenge distributions. Specifically, we show the following:

Theorem 6 (Informal)**.** *Suppose $\mathfrak{C}$ consists of polynomial-sized keyed circuits. Assuming the following:*

- *Post-quantum sub-exponentially secure indistinguishability obfuscation for* $\mathsf{P/poly}$*,*
- *Post-quantum sub-exponentially secure one-way functions,*
- *Learning with errors secure against QPT adversaries and,*
- *Simultaneous inner product conjecture.*

there exists generalized UPO *with respect to identical* $\mathcal{D}_{\mathcal{X}}$ *for* $\mathfrak{C}$*.*

On the Simultaneous Inner Product Conjecture: Technically we need two different versions of the simultaneous inner product conjecture (Conjecture 13 and Conjecture 14) to prove the security of our construction with respect to identical and independent challenge distributions. At a high level, the simultaneous inner product conjecture states that two (possibly entangled) QPT adversaries (i.e., non-local adversaries) should be unsuccessful in distinguishing $(\mathbf{r}, \langle \mathbf{r}, \mathbf{x} \rangle + m)$ versus $(\mathbf{r}, \langle \mathbf{r}, \mathbf{x} \rangle)$, where $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^n, \mathbf{x} \xleftarrow{\$} \mathbb{Z}_q^n, m \xleftarrow{\$} \mathbb{Z}_q$ for every prime $q \geq 1$. Moreover, the adversaries receive as input a bipartite state ρ that could depend on $\mathbf{x}$ with the guarantee that it should be computationally infeasible to recover $\mathbf{x}$. As mentioned above, we consider two different versions of the conjecture. In the first version (*identical*), both the adversaries get the same sample $(\mathbf{r}, \langle \mathbf{r}, \mathbf{x} \rangle)$ or they both get $(\mathbf{r}, \langle \mathbf{r}, \mathbf{x} \rangle + m)$. In the second version (*independent*), the main difference is that $\mathbf{r}$ and $\mathbf{x}$ are sampled independently for both adversaries. Weaker versions of this conjecture have been investigated and proven to be unconditionally true [AKL23, KT22].

Composition: Another contribution of ours is a composition theorem, where we show how to securely compose unclonable puncturable obfuscation with a functionality-preserving compiler. In more detail, we show the following. Suppose UPO is a secure unclonable puncturable obfuscation scheme and let $\mathsf{Compiler}$ be a functionality-preserving circuit compiler. We define another scheme UPO' such that the obfuscation algorithm of UPO', on input a circuit C, first runs the circuit compiler on C to obtain $\widetilde{C}$ and then it runs the obfuscation of UPO on $\widetilde{C}$ and outputs the result. The evaluation process can be similarly defined. We show that the resulting scheme UPO' is secure as long as UPO is secure. Our composition result allows us to compose UPO with other primitives such as different forms of program obfuscation without compromising on security. We use our composition theorem in some of the applications discussed earlier.

Concurrent and Independent Work. Concurrent to our work is a recent work by Coladangelo and Gunn [CG23] who also showed the feasibility of copy-protecting puncturable functionalities and point functions albeit using a completely different approach. At a high level, the themes of the two papers are quite different. Our goal is to identify a central primitive in unclonable cryptography whereas their work focuses on exploring applications of quantum state indistinguishability obfuscation, a notion of indistinguishability obfuscation for quantum computations, to unclonable cryptography.

We discuss the other differences below.

- Unlike our work, which only focuses on *search* puncturing security, their work considers both *search* and *decision* puncturing security.
- The two notions of obfuscation considered in both works seem to be incomparable. While the problem of obfuscating quantum computations has been notoriously challenging, their work considers the (weaker) problem of obfuscating a subclass of quantum computations that are implementations of classical functionalities.
- They demonstrate the feasibility of quantum state indistinguishability obfuscation in the quantum oracle model. We demonstrate the feasibility of UPO based on well-studied cryptographic assumptions and a new conjecture.

1.2 Technical Overview

We give an overview of the techniques behind our construction of UPO and the applications of UPO. We start with applications.

1.2.1 Applications

Copy-Protecting Puncturable Cryptographic Schemes. We begin by exploring methods to copy-protect puncturable pseudorandom functions. Subsequently, we generalize this approach to achieve copy-protection for a broader class of puncturable cryptographic schemes.

Case Study: Puncturable Pseudorandom Functions. Let $\mathcal{F} = \{f_k(\cdot) : \{0,1\}^n \to \{0,1\}^m : k \in \mathcal{K}_\lambda\}$ be a puncturable pseudorandom function (PRF) with λ being the security parameter and $\mathcal{K}_\lambda$ being the key space. To copy-protect $f_k(\cdot)$, we simply obfuscate $f_k(\cdot)$ using an unclonable puncturable obfuscation scheme UPO. To evaluate the copy-protected circuit on an input x, run the evaluation procedure of UPO.

To argue security, let us look at two experiments:

- The first experiment corresponds to the regular copy-protection security experiment. That is, $\mathcal{A}$ receives as input a copy-protected state ρ_{f_k}, which is copy-protection of f_k where k is sampled uniformly at random from the key space. It then creates a bipartite state which is split between $\mathcal{B}$ and $\mathcal{C}$, who are two non-communicating adversaries who can share some entanglement. Then, $\mathcal{B}$ and $\mathcal{C}$ independently receive as input x, which is picked uniformly at random. $(\mathcal{B}, \mathcal{C})$ win if they simultaneously guess $f_k(x)$.
- The second experiment is similar to the first experiment except $\mathcal{A}$ receives as input copy-protection of f_k punctured at the point x, where x is the same input given to both $\mathcal{B}$ and $\mathcal{C}$.

Thanks to the puncturing security of $\mathcal{F}$, the probability that $(\mathcal{B}, \mathcal{C})$ succeeds in the second experiment is negligible in λ. We would like to argue that $(\mathcal{B}, \mathcal{C})$ succeed in the first experiment also with probability negligible in λ. Suppose not, we show that the security of UPO is violated.

Reduction to UPO: The reduction $\mathcal{R}_\mathcal{A}$ samples a uniformly random f_k and forwards it to the challenger of the UPO game. The challenger of the UPO game then generates either an obfuscation of f_k or the punctured circuit f_k punctured at x which is then sent to $\mathcal{R}_\mathcal{A}$, who then forwards this to $\mathcal{A}$ who prepares the bipartite state. The reduction $\mathcal{R}_\mathcal{B}$ (resp., $\mathcal{R}_\mathcal{C}$) then receives as input x which it duly forwards to $\mathcal{B}$ (resp., $\mathcal{C}$). Then, $\mathcal{B}$ and $\mathcal{C}$ each output $y_\mathcal{B}$ and $y_\mathcal{C}$. Then, $\mathcal{R}_\mathcal{B}$ outputs the **bit 0** if $f_k(x) = y_\mathcal{B}$, otherwise it outputs 1. Similarly, $\mathcal{R}_\mathcal{C}$ outputs **bit 0** if $f_k(x) = y_\mathcal{C}$, otherwise it outputs 1. The reason behind boldifying "bit 0" part will be discussed below.

Let us see how $(\mathcal{R}_\mathcal{A}, \mathcal{R}_\mathcal{B}, \mathcal{R}_\mathcal{C})$ fares in the UPO game.

- *Case 1. Challenge bit is $b = 0$.* In this case, $\mathcal{R}_\mathcal{A}$ receives as input obfuscation of f_k with respect to UPO. Denote p_0 to be the probability that $(\mathcal{R}_\mathcal{B}, \mathcal{R}_\mathcal{C})$ output $(0, 0)$.
- *Case 2. Challenge bit is $b = 1$.* Here, $\mathcal{R}_\mathcal{A}$ receives as input obfuscation of the circuit f_k punctured at x. Similarly, denote p_1 to be the probability that $(\mathcal{R}_\mathcal{B}, \mathcal{R}_\mathcal{C})$ output $(1, 1)$.

From the security of UPO, we have the following: $\frac{p_0+p_1}{2} \leq \frac{1}{2} + \mu(\lambda)$, for some negligible function $\mu(\cdot)$. From the puncturing security of $\mathcal{F}$, the probability that $(\mathcal{R}_\mathcal{B}, \mathcal{R}_\mathcal{C})$ outputs $(1, 1)$ is at least $1 - \nu(\lambda)$, for some negligible function ν. In other words, $p_1 \geq 1 - \nu(\lambda)$. From this, we can conclude, p_0 is negligible which proves the security of the copy-protection scheme.

Perhaps surprisingly (at least to the authors), we do not know how to make the above reduction work if $\mathcal{R}_\mathcal{B}$ (resp., $\mathcal{R}_\mathcal{C}$) instead output bit 1 in the case when $f_k(x) = y_\mathcal{B}$ (resp., $f_k(x) = y_\mathcal{C}$). This is because we only get an upper bound for p_1 which cannot be directly used to determine an upper bound for p_0.

Generalizing to Puncturable Cryptographic Schemes. We present two generalizations of the above approach. We first generalize the above approach to handle puncturable circuit classes in Section 6.4. A circuit class $\mathfrak{C}$, equipped with an efficient puncturing algorithm Puncture, is said to be puncturable[3] if given a circuit $C \in \mathfrak{C}$, we can puncture C on a point x to obtain a punctured circuit G such that given punctured circuit G, it is computationally infeasible to predict $C(x)$. As we can see, puncturable pseudorandom functions are a special case of puncturable circuit classes. The template to copy-protect an arbitrary puncturable circuit class, say $\mathfrak{C}$, is essentially the same as the above template to copy-protect puncturable pseudorandom functions. To copy-protect C, obfuscate C using the scheme UPO. The evaluation process and the proof of security proceed along the same lines as above.

We then generalize this further to handle puncturable[4] cryptographic schemes. We consider an abstraction of a cryptographic scheme consisting of efficient algorithms (Gen, Eval, Puncture, Verify) with the following correctness guarantee: the verification algorithm on input (pk, x, y) outputs 1, where $\mathsf{Gen}(1^\lambda)$ produces the secret key-public key pair $(\mathsf{sk}, \mathsf{pk})$ and the value y is the output of Eval on input (sk, x). The algorithm Puncture on input (sk, x) outputs a punctured circuit that has the same functionality as $\mathsf{Eval}(\mathsf{sk}, \cdot)$ on all the points except x. The security property roughly states that predicting the output $\mathsf{Eval}(\mathsf{sk}, x)$ given the punctured circuit should be computationally infeasible. The above template of copy-protecting PRFs can similarly be adopted for copy-protecting puncturable cryptographic schemes.

Copy-Protecting Evasive Functions. Using UPO to construct copy-protection for evasive functions turns out to be more challenging. To understand the difficulty, let us compare both the

[3] We need a slightly more general version than this. Formally, in Definition 47, we puncture the circuit at two points (and not one), and then we require the adversary to predict the output of the circuit on one of the points.

[4] We again consider a more general version where the circuit is punctured at two points.

notions below:

- In a UPO scheme, $\mathcal{A}$ gets as input an obfuscation of a circuit C (if the challenge bit is $b = 0$) or a circuit C (if $b = 1$) punctured at two points $x^{\mathcal{B}}$ and $x^{\mathcal{C}}$. In the challenge phase, $\mathcal{B}$ gets $x^{\mathcal{B}}$ and $\mathcal{C}$ gets $x^{\mathcal{C}}$.

- In the copy-protection for evasive function scheme, $\mathcal{A}$ gets as input copy-protection of C, where C is a circuit implements an evasive function. In the challenge phase, $\mathcal{B}$ gets $x^{\mathcal{B}}$ and $\mathcal{C}$ gets $x^{\mathcal{C}}$, where $(x^{\mathcal{B}}, x^{\mathcal{C}}) = (x, x)$ is sampled as follows: x is sampled uniformly at random (if challenge bit is $b = 0$), otherwise x is sampled uniformly at random from the set of points on which C outputs 1 (if challenge bit is $b = 1$).

In other words, the distribution from which $\mathcal{A}$ gets its input from depends on the bit b in UPO but the challenges given to $\mathcal{B}$ and $\mathcal{C}$ are always sampled from the same distribution. The setting in the case of copy-protection is the opposite: the distribution from which $\mathcal{A}$ gets its input is always fixed while the challenge distribution depends on the bit b.

Preimage Sampling Property: To handle this discrepancy, we consider a class of evasive functions called preimage sampleable evasive functions. The first condition we require is that there is a distribution $\mathcal{D}$ from which we can efficiently sample a circuit C (representing an evasive function) together with an input x such that $C(x) = 1$. The second condition states that there exists another distribution $\mathcal{D}'$ from which we can sample (C', x'), where x' is sampled uniformly at random and then a punctured circuit C' is sampled conditioned on $C'(x') = 1$, satisfying the following property: the distributions $\mathcal{D}$ and $\mathcal{D}'$ are computationally indistinguishable. The second condition is devised precisely to ensure that we can reduce the security of copy-protection to UPO.

Construction and Proof Idea: But first let us discuss the construction of copy-protection: to copy-protect a circuit C, compute two layers of obfuscation of C. First, obfuscate C using a post-quantum iO scheme and then obfuscate the resulting circuit using UPO. To argue security, we view the obfuscated state given to $\mathcal{A}$ as follows: first sample C from $\mathcal{D}$ and then do the following: (a) give ρ_C to $\mathcal{A}$ if $b = 0$ and, (b) ρ_C to $\mathcal{A}$ if $b = 1$, where ρ_C is the copy-protected state and b is the challenge bit that is used in the challenge phase. So far, we have done nothing. Now, we will modify (b). We will leverage the above conditions to modify (b) as follows: we will instead sample from $\mathcal{D}'$. Since $\mathcal{D}$ and $\mathcal{D}'$ are computationally indistinguishable, the adversary will not notice the change. Now, let us examine the modified experiment: if $b = 0$, the adversary receives ρ_C (defined above), where (C, x) is sampled from $\mathcal{D}$ and if $b = 1$, the adversary receives $\rho_{C'}$, where (C', x') is sampled from $\mathcal{D}'$. We can show that this precisely corresponds to the UPO experiment and thus, we can successfully carry out the reduction.

Single-Decryptor Encryption. A natural attempt to construct single-decryptor encryption would be to leverage UPO for puncturable cryptographic schemes. After all, it would seem that finding a public-key encryption scheme where the decryption key can be punctured at the challenge ciphertexts would give us our desired result. Unfortunately, this does not quite work: the reason lies in the way we defined the challenge distribution of UPO. We required that the marginals of the challenge distribution for a UPO scheme have to be uniform. Any public-key encryption scheme where the decryption keys can be punctured would not necessarily satisfy this requirement and hence, we need to find schemes that do[5].

We start with the public-key encryption scheme due to Sahai and Waters [SW14]. The advantage of this scheme is that the ciphertexts are pseudorandom. First, we show that this public-key encryption scheme can be made puncturable. Once we show this, using UPO for puncturable cryptographic schemes (and standard iO tricks), we construct single-decryptor encryption schemes of two flavors:

- First, we consider search security (Figure 32). In this security definition, $\mathcal{B}$ and $\mathcal{C}$ receive ciphertexts of random messages and they win if they are able to predict the messages.
- Next, we consider selective security (Figure 35). In this security definition, $\mathcal{B}$ and $\mathcal{C}$ receive encryptions of one of two messages adversarially chosen and they are supposed to predict which of the two messages was used in the encryption. Moreover, the adversarially chosen messages need to be declared before the security experiment begins and hence, the term selective security. Once we achieve this, we propose a generic lifting theorem to lift SDE security satisfying selective security to full adaptive security (Figure 36) where the challenge messages can be chosen later in the experiment.

1.2.2 Construction of UPO

We move on to the construction of UPO.

Starting Point: Decoupling Unclonability and Computation. We consider the following template to design UPO. To obfuscate a circuit C, we build two components. The first component is an unclonable quantum state that serves the purpose of authentication. The second component is going to aid in computation once the authentication passes. In more detail, given an input x, we first use the unclonable quantum state to authenticate x and then execute the second component on the authenticated tag along with x to obtain the output $C(x)$.

The purpose of designing the obfuscation scheme this way is two-fold. Firstly, the fact that the first component is an unclonable quantum state means that an adversary cannot create multiple copies of this. And by design, without this state, it is not possible to execute the second component. Secondly, decoupling the unclonability and the computation part allows us to put less burden on the unclonable state, and in particular, only require the first component for authentication. Moreover, this approach helps us leverage existing tools in a modular way to construct UPO.

To implement the above approach, we use a copy-protection scheme for pseudorandom functions [CLLZ21], denoted by CP, and a post-quantum indistinguishability obfuscation scheme, denoted by iO. In the UPO scheme, to obfuscate C, we do the following:

1. Copy-protect a pseudorandom function $f_k(\cdot)$ and,
2. Obfuscate a circuit, with the PRF key k hardcoded in it, that takes as input (x, y) and outputs $C(x)$ if and only if $f_k(x) = y$.

First Issue. While syntactically the above template makes sense, when proving security we run into an issue. To invoke the security of CP, we need to argue that the obfuscated circuit does not

[5] Of course, we could try the aforementioned issue in a different way: we could instead relax the requirements on the challenge distribution of UPO. Unfortunately, we currently do not know how to design an UPO for challenge distributions that do not have uniform marginals.

reveal any information about the PRF key k. This suggests that we need a much stronger object like virtual black box obfuscation instead of iO which is in general known to be impossible [BGI+01]. Taking a closer look, we realize that this issue arose because we wanted to completely decouple the CP part and the iO part.

Second Issue. Another issue that arises when attempting to work out the proof. At a high level, in the security proof, we reach a hybrid where we need to hardwire the outputs of the PRF on the challenge inputs $x^{\mathcal{B}}$ and $x^{\mathcal{C}}$ in the obfuscated circuit (i.e., in bullet 2 above). This creates an obstacle when we need to invoke the security of copy-protection: the outputs of the PRF are only available in the challenge phase (i.e., *after* $\mathcal{A}$ splits) whereas we need to know these outputs in order to generate the input to $\mathcal{A}$.

Addressing the above issues. We first address the second issue. We introduce a new security notion of copy-protection for PRFs, referred to as copy-protection with *preponed security*. Roughly speaking, in the preponed security experiment, $\mathcal{A}$ receives the outputs of the PRF on the challenge inputs instead of being delayed until the challenge phase. By design, this stronger security notion solves the second issue.

In order to resolve the aforementioned problem, we pull back and only partially decouple the two components. In particular, we tie both the CP and iO parts together by making non-black-box use of the underlying copy-protection scheme. Specifically, we rely upon the scheme by Colandangelo et al. [CLLZ21]. Moreover, we show that Colandangelo et al. [CLLZ21] scheme satisfies preponed security by reducing their security to the security of their single-decryptor encryption construction; our proof follows along the same lines as theirs. Unfortunately, we do not know how to go further. While they did show that their single-decryptor encryption construction can be based on well studied cryptographic assumptions, the type of single-decryptor encryption scheme we need has a different flavor. In more detail, in their scheme, they consider *independent* challenge distribution (i.e., both $\mathcal{B}$ and $\mathcal{C}$ receive ciphertexts where the challenge bit is picked independently), whereas we consider *identical* challenge distribution (i.e., the challenge bit for both $\mathcal{B}$ and $\mathcal{C}$ is identical). We show how to modify their construction to satisfy security with respect to identical challenge distribution based on the simultaneous inner product conjecture.

Summary. To summarise, we design UPO for keyed circuit classes in P/poly as follows:

- We show that as long as the copy-protection scheme of [CLLZ21] satisfies preponed security, UPO for P/poly exists. This step makes heavy use of iO techniques.
- We reduce the task of proving that the copy-protection scheme of [CLLZ21] satisfies preponed security to the task of proving that the single-decryptor encryption construction of [CLLZ21] is secure in the identical challenge setting.

2 Preliminaries

We refer the reader to [NC10] for a comprehensive reference on the basics of quantum information and quantum computation. We use I to denote the identity operator. We use $\mathcal{S}(\mathcal{H})$ to denote the set of unit vectors in the Hilbert space $\mathcal{H}$. We use $\mathcal{D}(\mathcal{H})$ to denote the set of density matrices in the Hilbert space $\mathcal{H}$. Let P, Q be distributions. We use $d_{TV}(P, Q)$ to denote the total variation distance between them. Let $\rho, \sigma \in \mathcal{D}(\mathcal{H})$ be density matrices. We write $\mathsf{TD}(\rho, \sigma)$ to denote the trace distance between them, i.e.,

$$\mathsf{TD}(\rho, \sigma) = \frac{1}{2}\|\rho - \sigma\|_1$$

where $\|\|X\|\|_1 = \mathsf{Tr}(\sqrt{X^\dagger X})$ denotes the trace norm. We denote $\|\|X\|\| := \sup_{|\psi\rangle}\{\langle\psi|X|\psi|\}\rangle$ to be the operator norm where the supremum is taken over all unit vectors. For a vector $|x\rangle$, we denote its Euclidean norm to be $\|\||x\rangle\|\|_2$. We use the notation $M \geq 0$ to denote the fact that M is positive semi-definite.

2.1 Quantum Algorithms

A quantum algorithm A is a family of generalized quantum circuits $\{A_\lambda\}_{\lambda\in\mathbb{N}}$ over a discrete universal gate set (such as $\{CNOT, H, T\}$). By generalized, we mean that such circuits can have a subset of input qubits that are designated to be initialized in the zero state and a subset of output qubits that are designated to be traced out at the end of the computation. Thus a generalized quantum circuit A_λ corresponds to a *quantum channel*, which is a completely positive trace-preserving (CPTP) map. When we write $A_\lambda(\rho)$ for some density matrix ρ, we mean the output of the generalized circuit A_λ on input ρ. If we only take the quantum gates of A_λ and ignore the subset of input/output qubits that are initialized to zeroes/traced out, then we get the *unitary part* of A_λ, which corresponds to a unitary operator which we denote by $\hat{A}_\lambda$. The *size* of a generalized quantum circuit is the number of gates in it, plus the number of input and output qubits.

We say that $A = \{A_\lambda\}_\lambda$ is a *quantum polynomial-time (QPT) algorithm* if there exists a polynomial p such that the size of each circuit A_λ is at most $p(\lambda)$. We furthermore say that A is *uniform* if there exists a deterministic polynomial-time Turing machine M that on input 1^λ outputs the description of A_λ.

We also define the notion of a *non-uniform* QPT algorithm A that consists of a family $\{(A_\lambda, \rho_\lambda)\}_\lambda$ where $\{A_\lambda\}_\lambda$ is a polynomial-size family of circuits (not necessarily uniformly generated), and for each λ there is additionally a subset of input qubits of A_λ that are designated to be initialized with the density matrix ρ_λ of polynomial length. This is intended to model nonuniform quantum adversaries who may receive quantum states as advice. Nevertheless, the reductions we show in this work are all uniform.

The notation we use to describe the inputs/outputs of quantum algorithms will largely mimic what is used in the classical cryptography literature. For example, for a state generator algorithm G, we write $G_\lambda(k)$ to denote running the generalized quantum circuit G_λ on input $|k\rangle\langle k|$, which outputs a state ρ_k.

Ultimately, all inputs to a quantum circuit are density matrices. However, we mix-and-match between classical, pure state, and density matrix notation; for example, we may write $A_\lambda(k, |\theta\rangle, \rho)$ to denote running the circuit A_λ on input $|k\rangle\langle k| \otimes |\theta\rangle\langle\theta| \otimes \rho$. In general, we will not explain all the input and output sizes of every quantum circuit in excruciating detail; we will implicitly assume that a quantum circuit in question has the appropriate number of input and output qubits as required by the context.

3 Unclonable Puncturable Obfuscation: Definition

We present the definition of an unclonable puncturable obfuscation scheme in this section.

Keyed Circuit Class. A class of classical circuits of the form $\mathfrak{C} = \{\mathfrak{C}_\lambda\}_{\lambda \in \mathbb{N}}$ is said to be a keyed circuit class if the following holds: $\mathfrak{C}_\lambda = \{C_k : k \in \mathcal{K}_\lambda\}$, where C_k is a (classical) circuit with input length $n(\lambda)$, output length $m(\lambda)$ and $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda \in \mathbb{N}}$ is the key space. We refer to C_k as a keyed circuit. We note that any circuit class can be represented as a keyed circuit class using universal circuits. We will be interested in the setting when C_k is a polynomial-sized circuit; henceforth, unless specified otherwise, all keyed circuit classes considered in this work will consist only of polynomial-sized circuits. We will also make a simplifying assumption that C_k and $C_{k'}$ have the same size, where $k, k' \in \mathcal{K}_\lambda$.

Syntax. An unclonable puncturable obfuscation (UPO) scheme $(\mathsf{Obf}, \mathsf{Eval})$ for a keyed circuit class $\mathfrak{C} = \{\mathfrak{C}_\lambda\}_{\lambda \in \mathbb{N}}$, consists of the following QPT algorithms:

- $\mathsf{Obf}(1^\lambda, C)$: on input a security parameter λ and a keyed circuit $C \in \mathfrak{C}_\lambda$ with input length $n(\lambda)$, it outputs a quantum state ρ_C.
- $\mathsf{Eval}(\rho_C, x)$: on input a quantum state ρ_C and an input $x \in \{0,1\}^{n(\lambda)}$, it outputs (ρ'_C, y).

Correctness. An unclonable puncturable obfuscation scheme $(\mathsf{Obf}, \mathsf{Eval})$ for a keyed circuit class $\mathfrak{C} = \{\mathfrak{C}_\lambda\}_{\lambda \in \mathbb{N}}$ is δ-correct, if for every $C \in \mathfrak{C}_\lambda$ with input length $n(\lambda)$, and for every $x \in \{0,1\}^{n(\lambda)}$,

$$\Pr\left[C(x) = y \;\middle|\; \substack{\rho_C \leftarrow \mathsf{Obf}(1^\lambda, C) \\ (\rho'_C, y) \leftarrow \mathsf{Eval}(\rho_C, x)}\right] \geq \delta$$

If δ is negligibly close to 1 then we say that the scheme is correct (i.e., we omit mentioning δ).

Remark 7. *If $(1 - \delta)$ is a negligible function in λ, by invoking the almost as good as new lemma [Aar16], we can evaluate ρ'_C on another input x' to get $C(x')$ with probability negligibly close to 1. We can repeat this process polynomially many times and each time, due to the quantum union bound [Gao15], we get the guarantee that the output is correct with probability negligibly close to 1.*

3.1 Security

Puncturable Keyed Circuit Class. Consider a keyed circuit class $\mathfrak{C} = \{\mathfrak{C}_\lambda\}_{\lambda \in \mathbb{N}}$, where $\mathfrak{C}_\lambda$ consists of circuits of the form $C_k(\cdot)$, where $k \in \mathcal{K}_\lambda$, the input length of $C_k(\cdot)$ is $n(\lambda)$ and the output length is $m(\lambda)$. We say that $\mathfrak{C}_\lambda$ is said to be puncturable if there exists a deterministic polynomial-time puncturing algorithm $\mathsf{Puncture}$ such that the following holds: on input $k \in \{0,1\}^\lambda$, strings $x^{\mathcal{B}} \in \{0,1\}^{n(\lambda)}, x^{\mathcal{C}} \in \{0,1\}^{n(\lambda)}$, it outputs a circuit G_{k^*}. Moreover, the following holds: for every $x \in \{0,1\}^{n(\lambda)}$,

$$G_{k^*}(x) = \begin{cases} C_k(x), & x \neq x^{\mathcal{B}}, x \neq x^{\mathcal{C}}, \\ \perp, & x \in \{x^{\mathcal{B}}, x^{\mathcal{C}}\}. \end{cases}$$

Without loss of generality, we can assume that the size of G_{k^*} is the same as the size of C_k.

Definition 8 (UPO Security)**.** *We say that a pair of QPT algorithms* $(\mathsf{Obf}, \mathsf{Eval})$ *for a puncturable keyed circuit class $\mathfrak{C}$, associated with puncturing procedure* $\mathsf{Puncture}$, *satisfies* **UPO security** *with respect to a distribution $\mathcal{D}_{\mathcal{X}}$ on $\{0,1\}^{n(\lambda)} \times \{0,1\}^{n(\lambda)}$ if the following holds for every QPT $(\mathcal{A}, \mathcal{B}, \mathcal{C})$:*

$$\mathsf{Pr}\left[1 \leftarrow \mathsf{UPO.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}_{\mathcal{X}},\mathfrak{C}}\left(1^\lambda, b\right) \;:\; b \xleftarrow{\$} \{0,1\}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

where $\mathsf{UPO.Expt}$ *is defined in Figure 2.*

$\underline{\mathsf{UPO.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}_{\mathcal{X}},\mathfrak{C}}\left(1^{\lambda},b\right)}$:

- $\mathcal{A}$ sends k, where $k \in \mathcal{K}_\lambda$, to the challenger Ch.
- Ch samples $(x^{\mathcal{B}}, x^{\mathcal{C}}) \leftarrow \mathcal{D}_{\mathcal{X}}(1^\lambda)$ and generates $G_{k^*} \leftarrow \mathsf{Puncture}(k, x^{\mathcal{B}}, x^{\mathcal{C}})$.
- Ch generates ρ_b as follows:
  - $\rho_0 \leftarrow \mathsf{Obf}(1^\lambda, C_k(\cdot))$,
  - $\rho_1 \leftarrow \mathsf{Obf}(1^\lambda, G_{k^*}(\cdot))$

  It sends ρ_b to $\mathcal{A}$.
- Apply $(\mathcal{B}(x^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(x^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_{\mathbf{B}}, b_{\mathbf{C}})$.
- Output 1 if $b = b_{\mathbf{B}} = b_{\mathbf{C}}$.

Figure 2: Security Experiment

3.1.1 Generalized Security

For most applications, the security definition discussed in Section 3.1 suffices. But for a couple of applications, we need a generalized definition. The new definition generalizes the definition in Section 3.1 in terms of puncturability as follows. We allow the adversary to choose the outputs of the circuit generated by $\mathsf{Puncture}$ on the punctured points. Previously, the circuit generated by the puncturing algorithm was such that on the punctured points, it output $\perp$. Instead, we allow the adversary to decide the values that need to be output on the points that are punctured. We emphasize that the adversary still would not know the punctured points itself until the challenge phase. Formally, the (generalized) puncturing algorithm $\mathsf{GenPuncture}$ now takes as input $k \in \mathcal{K}_\lambda$, polynomial-sized circuits $\mu^{\mathcal{B}} : \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}$, $\mu^{\mathcal{C}} : \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}$, strings $x^{\mathcal{B}} \in \{0,1\}^{n(\lambda)}, x^{\mathcal{C}} \in \{0,1\}^{n(\lambda)}$, if $x^{\mathcal{B}} \neq x^{\mathcal{C}}$, it outputs a circuit G_{k^*} such that for every $x \in \{0,1\}^{n(\lambda)}$,

$$G_{k^*}(x) = \begin{cases} C_k(x), & x \neq x^{\mathcal{B}}, x \neq x^{\mathcal{C}} \\ \mu_{\mathcal{B}}(x^{\mathcal{B}}), & x = x^{\mathcal{B}} \\ \mu_{\mathcal{C}}(x^{\mathcal{C}}), & x = x^{\mathcal{C}}, \end{cases}$$

else it outputs a circuit G_{k^*} such that for every $x \in \{0,1\}^{n(\lambda)}$,

$$G_{k^*}(x) = \begin{cases} C_k(x), & x \neq x^{\mathcal{B}} \\ \mu_{\mathcal{B}}(x^{\mathcal{B}}), & x = x^{\mathcal{B}}. \end{cases}$$

As before, we assume that without loss of generality, the size of G_{k^*} is the same as the size of C_k.

A keyed circuit class $\mathfrak{C}$ associated with a generalized puncturing algorithm $\mathsf{GenPuncture}$ is referred to as a *generalized puncturable keyed circuit class.*

Definition 9 (Generalized UPO security). *We say that a pair of QPT algorithms* ($\mathsf{Obf}, \mathsf{Eval}$) *for a generalized keyed circuit class* $\mathfrak{C} = \{\mathfrak{C}_\lambda\}_{\lambda \in \mathbb{N}}$ *equipped with a puncturing algorithm* $\mathsf{GenPuncture}$*, satisfies* **generalized UPO security** *with respect to a distribution* $\mathcal{D}_\mathcal{X}$ *on* $\{0,1\}^{n(\lambda)} \times \{0,1\}^{n(\lambda)}$ *if the following holds for every QPT* $(\mathcal{A}, \mathcal{B}, \mathcal{C})$*:*

$$\Pr\left[1 \leftarrow \mathsf{GenUPO.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}_\mathcal{X},\mathfrak{C}}\left(1^\lambda, b\right) \ : \ b \xleftarrow{\$} \{0,1\}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

where $\mathsf{GenUPO.Expt}$ *is defined in Figure 3.*

$\underline{\mathsf{GenUPO.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}_\mathcal{X},\mathfrak{C}}\left(1^\lambda, b\right)}$:

- $\mathcal{A}$ sends $(k, \mu_\mathcal{B}, \mu_\mathcal{C})$, where $k \in \mathcal{K}_\lambda, \mu_\mathcal{B} : \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}, \mu_\mathcal{C} : \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}$, to the challenger Ch.
- Ch samples $(x^\mathcal{B}, x^\mathcal{C}) \leftarrow \mathcal{D}_\mathcal{X}(1^\lambda)$ and generates $G_{k^*} \leftarrow \mathsf{Puncture}(k, x^\mathcal{B}, x^\mathcal{C}, \mu_\mathcal{B}, \mu_\mathcal{C})$.
- Ch generates ρ_b as follows:
  - $\rho_0 \leftarrow \mathsf{Obf}(1^\lambda, C_k)$,
  - $\rho_1 \leftarrow \mathsf{Obf}(1^\lambda, G_{k^*})$

  It sends ρ_b to $\mathcal{A}$.
- Apply $(\mathcal{B}(x^\mathcal{B}, \cdot) \otimes \mathcal{C}(x^\mathcal{C}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_\mathbf{B}, b_\mathbf{C})$.
- Output 1 if $b = b_\mathbf{B} = b_\mathbf{C}$.

Figure 3: Generalized Security Experiment

Instantiations of $\mathcal{D}_\mathcal{X}$. In the applications, we will be considering the following two distributions:

1. $\mathcal{U}_{\{0,1\}^{2n}}$ß: the uniform distribution on $\{0,1\}^{2n}$. When the context is clear, we simply refer to this distribution as $\mathcal{U}$.
2. $\mathsf{Id}_\mathcal{U}\{0,1\}^n$: identical distribution on $\{0,1\}^n \times \{0,1\}^n$ with uniform marginals. That is, the sampler for $\mathsf{Id}_\mathcal{U}\{0,1\}^n$ is defined as follows: sample x from $\mathcal{U}_{\{0,1\}^n}$ and output (x, x).

3.2 Composition Theorem

We state a useful theorem that states that we can compose a secure UPO scheme with any functionality-preserving compiler without compromising on security.

Let $\mathsf{Compile}$ be a circuit compiler, i.e., $\mathsf{Compile}$ is a probabilistic algorithm that takes as input a security parameter λ, classical circuit C and outputs another classical circuit $\widetilde{C}$ such that C and $\widetilde{C}$ have the same functionality. For instance, program obfuscation [BGI+01] is an example of a circuit compiler.

Let $\mathfrak{C}$ be a generalized puncturable keyed circuit class associated with keyspace $\mathcal{K}$ defined as follows: $\mathfrak{C} = \{\mathfrak{C}_\lambda\}_{\lambda\in\mathbb{N}}$, where every circuit in $\mathfrak{C}_\lambda$ is of the form C_k, where $k \in \mathcal{K}_\lambda$, with input length $n(\lambda)$ and the output length $m(\lambda)$. We denote $\mathsf{GenPuncture}$ to be a generalized puncturing algorithm associated with $\mathfrak{C}$.

Let $\mathsf{UPO} = (\mathsf{UPO.Obf}, \mathsf{UPO.Eval})$ be an unclonable puncturable obfuscation scheme for a generalized puncturable keyed circuit class $\mathfrak{G}$ (defined below) with respect to the input distribution $\mathcal{D}_{\mathcal{X}}$.

We define $\mathfrak{G} = \{\mathfrak{G}_\lambda\}_{\lambda\in\mathbb{N}}$, where every circuit in $\mathfrak{G}_\lambda$ is of the form $G_{k||r}(\cdot)$, with input length $n(\lambda)$, output length $m(\lambda)$, $k \in \mathcal{K}_\lambda$, and $r \in \{0,1\}^{t(\lambda)}$. Here, $t(\lambda)$ denotes the number of bits of randomness consumed by $\mathsf{Compile}(1^\lambda, C_k; \cdot)$. Moreover, the circuit $G_{k||r}$ takes as input $x \in \{0,1\}^n$, applies $\mathsf{Compile}(1^\lambda, C_k; r)$ to obtain $\widetilde{C_k}$ and then it outputs $\widetilde{C_k}(x)$. The puncturing algorithm associated with $\mathfrak{G}$ is $\mathsf{GenPuncture}'$ which on input $k\|r$ and the set of inputs x_1, x_2 and circuits μ_1, μ_2, generates $D_{k^*} \leftarrow \mathsf{GenPuncture}(k, x_1, x_2, \mu_1, \mu_2)$, and then outputs the circuit $G_{k^*,r}$, where $G_{k^*,r}$ is defined as follows: it takes as input $x \in \{0,1\}^n$, applies $\mathsf{Compile}(1^\lambda, D_{k^*}; r)$ to obtain $\widetilde{D_{k^*}}$ and then it outputs $\widetilde{D_{k^*}}$. The keyspace associated with $\mathfrak{G}$ is $\mathcal{K}' = \{\mathcal{K}'_\lambda\}_{\lambda\in\mathbb{N}}$, where $\mathcal{K}'_\lambda = \mathcal{K}_\lambda \times \{0,1\}^{t(\lambda)}$.

We define $\mathsf{UPO}' = (\mathsf{UPO}'.\mathsf{Obf}, \mathsf{UPO}'.\mathsf{Eval})$ as follows:

- $\mathsf{UPO}'.\mathsf{Obf}(1^\lambda, C) = \mathsf{UPO.Obf}(1^\lambda, \widetilde{C})$, where $\widetilde{C} \leftarrow \mathsf{Compile}(1^\lambda, C)$.
- $\mathsf{UPO}'.\mathsf{Eval} = \mathsf{UPO.Eval}$.

Proposition 10. *Assuming* UPO *satisfies* $\mathcal{D}_{\mathcal{X}}$*-generalized unclonable puncturable obfuscation security for* $\mathfrak{G}$ *and* $\mathsf{Compile}$ *is a circuit compiler for* $\mathfrak{C}$*,* UPO' *satisfies* $\mathcal{D}_{\mathcal{X}}$*-generalized unclonable puncturable obfuscation security for* $\mathfrak{C}$*.*

Proof. Suppose there is an adversary $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ that violates the security of UPO' with probability p. We construct a QPT reduction $(\mathcal{R}_\mathcal{A}, \mathcal{R}_\mathcal{B}, \mathcal{R}_\mathcal{C})$ that violates the security of UPO, also with probability p. From the security of UPO' it then follows that p is at most $\frac{1}{2} + \varepsilon$, for some negligible function ε, which proves the theorem.

$\mathcal{R}_\mathcal{A}(1^\lambda)$ first runs $\mathcal{A}(1^\lambda)$ to obtain $k \in \mathcal{K}_\lambda$. It then samples $r \xleftarrow{\$} \{0,1\}^{t(\lambda)}$. Then, $\mathcal{R}_\mathcal{A}$ forwards $k||r$ to the external challenger of UPO. Then, $\mathcal{R}_\mathcal{A}$ receives ρ^* which it then duly forwards to $\mathcal{A}$. Similarly, even in the challenge phase, $\mathcal{R}_\mathcal{B}$ (resp., $\mathcal{R}_\mathcal{C}$) forwards the challenge from the challenger to $\mathcal{B}$ (resp., $\mathcal{C}$).

It can be seen that the probability that $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ breaks the security of UPO' is the same as the probability that $(\mathcal{R}_\mathcal{A}, \mathcal{R}_\mathcal{B}, \mathcal{R}_\mathcal{C})$ breaks the security of UPO. □

Theorem 11 (Composition theorem)**.** *Let* $\mathsf{Compile}$ *be a circuit compiler, i.e.,* $\mathsf{Compile}$ *is a probabilistic algorithm that takes as input a classical circuit* C *and outputs another classical circuit* $\tilde{C}$ *such that* C *and* $\tilde{C}$ *have the same functionality. Let* $\mathsf{UPO} = (\mathsf{UPO.Obf}, \mathsf{UPO.Eval})$ *be an unclonable* *puncturable obfuscation scheme that satisfies $\mathcal{D}_{\mathcal{X}}$-generalized unclonable puncturable obfuscation security for any class of generalized puncturable keyed circuit classs in* $\mathsf{P/poly}$*, then the same holds for the unclonable puncturable obfuscation scheme* $\mathsf{UPO}' = (\mathsf{UPO}'.\mathsf{Obf}, \mathsf{UPO}'.\mathsf{Eval})$ *defined as follows:*

- $\mathsf{UPO}'.\mathsf{Obf}(1^\lambda, C) = \mathsf{UPO}.\mathsf{Obf}(1^\lambda, \mathsf{Compile}(C))$ *for every circuit* C.
- $\mathsf{UPO}'.\mathsf{Eval} = \mathsf{UPO}.\mathsf{Eval}$.

Proof. Let $\mathfrak{C}$ be an arbitrary generalized puncturable keyed class in $\mathsf{P/poly}$. Let $\mathfrak{G}$ be the generalized puncturable keyed class in $\mathsf{P/poly}$ derived from $\mathfrak{C}$ as defined on Page 18. Note that by the assumption in the theorem, UPO satisfies $\mathcal{D}_{\mathcal{X}}$-generalized unclonable puncturable obfuscation security for $\mathfrak{G}$. Therefore, by Proposition 10, UPO' satisfies $\mathcal{D}_{\mathcal{X}}$-generalized unclonable puncturable obfuscation security for $\mathfrak{C}$. Since $\mathfrak{C}$ was arbitrary, we conclude that UPO' satisfies $\mathcal{D}_{\mathcal{X}}$-generalized unclonable puncturable obfuscation security for any generalized puncturable keyed circuit class in $\mathsf{P/poly}$. □

Instantiating $\mathsf{Compile}$ with an indistinguishability obfuscation iO in theorem 11, the following corollary is immediate.

Corollary 12. *Consider a keyed circuit class* $\mathfrak{C}$*. Suppose* iO *be an indistinguishability obfuscation scheme for* $\mathfrak{C}$*. Suppose* UPO *is an unclonable puncturable obfuscation scheme for* $\mathfrak{G}$ *(as defined above). Then* UPO' *is a secure unclonable puncturable obfuscation scheme for* $\mathfrak{C}$ *where* UPO' *is defined as follows:*

Assuming UPO *is a unclonable puncturable obfuscation scheme that satisfies* $\mathcal{D}_{\mathcal{X}}$*-generalized unclonable puncturable obfuscation security for any* $\mathcal{D}_{\mathcal{X}}$*-generalized puncturable keyed circuit class in* $\mathsf{P/poly}$*, then the same holds for the unclonable puncturable obfuscation scheme* $\mathsf{UPO}' = (\mathsf{UPO}'.\mathsf{Obf}, \mathsf{UPO}'.\mathsf{Eval})$ *defined as follows:*

- $\mathsf{UPO}'.\mathsf{Obf}(1^\lambda, C) = \mathsf{UPO}.\mathsf{Obf}(1^\lambda, \mathsf{iO}(1^\lambda, C))$*, where* $C \in \mathfrak{C}_\lambda$.
- $\mathsf{UPO}'.\mathsf{Eval} = \mathsf{UPO}.\mathsf{Eval}$.

In the corollary above, we assume that the indistinguishability scheme does not have an explicit evaluation algorithm. In other words, the obfuscation algorithm on input a circuit C outputs another circuit $\widetilde{C}$ that is functionally equivalent to C. This is without loss of generality since we can combine any indistinguishability obfuscation scheme (that has an evaluation algorithm) with universal circuits to obtain an obfuscation scheme with the desired format.

4 Conjectures

The security of our construction relies upon some novel conjectures. Towards understanding our conjectures, consider the following problem: suppose say an adversary $\mathcal{B}$ is given a state $\rho_{\mathbf{x}}$ that is generated as a function of a secret string $\mathbf{x} \in \mathbb{Z}_q^n$, where $q, n \in \mathbb{N}$ and q is prime. We are given the guarantee that just given $\rho_{\mathbf{x}}$, it should be infeasible to compute $\mathbf{x}$ for most values of $\mathbf{x}$. Now, the goal of $\mathcal{B}$ is to distinguish $(\mathbf{u}, \langle \mathbf{u}, \mathbf{x} \rangle)$, where $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^n$ versus $(\mathbf{u}, \mathbf{x}) + m$, where $m \xleftarrow{\$} \mathbb{Z}_q$. The Goldreich-Levin precisely shows that $\mathcal{B}$ cannot succeed; if $\mathcal{B}$ did succeed then we can come up with an extractor that recovers $\mathbf{x}$. Our conjectures state that the problem should be hard even for two (possibly entangled) parties simultaneously distinguishing the above samples. Depending on whether the samples are independently generated between these parties or they are correlated, we have two different conjectures.

Before we formally state these conjectures and prove them, we first define the following problem.

$(\mathcal{D}_{\mathcal{X}}, \mathcal{D}_{\mathsf{Ch}}, \mathcal{D}_{\mathsf{bit}})$-Simultaneous Inner Product Problem ($(\mathcal{D}_{\mathcal{X}}, \mathcal{D}_{\mathsf{Ch}}, \mathcal{D}_{\mathsf{bit}})$-simultIP). Let $\mathcal{D}_{\mathcal{X}}$ be a distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q^n$, $\mathcal{D}_{\mathsf{Ch}}$ be a distribtion on $\mathbb{Z}_q^{n+1} \times \mathbb{Z}_q^{n+1}$ and finally, let $\mathcal{D}_{\mathsf{bit}}$ be a distribution on $\{0,1\} \times \{0,1\}$, for some $q \in \mathbb{N}$. Let $\mathcal{B}'$ and $\mathcal{C}'$ be QPT algorithms. Let $\rho = \{\rho_{\mathbf{x}^{\mathcal{B}},\mathbf{x}^{\mathcal{C}}}\}_{\mathbf{x}^{\mathcal{B}},\mathbf{x}^{\mathcal{C}} \in \mathbb{Z}_q^n}$ be a set of bipartite states. Consider the following game.

- Sample $(\mathbf{x}^{\mathcal{B}}, \mathbf{x}^{\mathcal{C}}) \leftarrow \mathcal{D}_{\mathcal{X}}$.
- Sample $\left((\mathbf{u}^{\mathcal{B}}, m^{\mathcal{B}}), (\mathbf{u}^{\mathcal{C}}, m^{\mathcal{C}})\right) \leftarrow \mathcal{D}_{\mathsf{Ch}}$
- Set $z_0^{\mathcal{B}} = \langle \mathbf{u}^{\mathcal{B}}, \mathbf{x}^{\mathcal{B}} \rangle, z_0^{\mathcal{C}} = \langle \mathbf{u}^{\mathcal{C}}, \mathbf{x}^{\mathcal{C}} \rangle, z_1^{\mathcal{B}} = m^{\mathcal{B}} + \langle \mathbf{u}^{\mathcal{B}}, \mathbf{x}^{\mathcal{B}} \rangle, z_1^{\mathcal{C}} = m^{\mathcal{C}} + \langle \mathbf{u}^{\mathcal{C}}, \mathbf{x}^{\mathcal{C}} \rangle$
- Sample $(b^{\mathcal{B}}, b^{\mathcal{C}}) \leftarrow \mathcal{D}_{\mathsf{bit}}$
- $(\widehat{b}^{\mathcal{B}}, \widehat{b}^{\mathcal{C}}) \leftarrow (\mathcal{B}'(\mathbf{u}^{\mathcal{B}}, z_{b^{\mathcal{B}}}^{\mathcal{B}}, \cdot) \otimes \mathcal{C}'(\mathbf{u}^{\mathcal{C}}, z_{b^{\mathcal{C}}}^{\mathcal{C}}, \cdot))(\rho_{\mathbf{x}^{\mathcal{B}},\mathbf{x}^{\mathcal{C}}})$

We say that $(\mathcal{B}', \mathcal{C}')$ succeeds if $\widehat{b}^{\mathcal{B}} = b^{\mathcal{B}}$ and $\widehat{b}^{\mathcal{C}} = b^{\mathcal{C}}$.

Specific Settings. Consider the following setting: (a) $q = 2$, (b) $\mathcal{D}_{\mathsf{bit}}$ is a uniform distribution on $\{0,1\}^2$, (c) $\mathcal{D}_{\mathsf{Ch}}$ is a uniform distribution on $\mathbb{Z}_q^{2n+2}$ and $\mathcal{D}_{\mathcal{X}}$ is a uniform distribution on $\{(\mathbf{x}, \mathbf{x}) : \mathbf{x} \in \mathbb{Z}_q^n\}$. In this setting, recent works [KT22, AKL23] showed, via a simultaneous version of quantum Goldreich-Levin theorem, that any non-local solver for the $(\mathcal{D}_{\mathsf{Ch}}, \mathcal{D}_{\mathsf{bit}})$-simultaneous inner product problem can succeed with probability at most $\frac{1}{2} + \varepsilon(n)$, for some negligible function $\varepsilon(n)$. Although not explicitly stated, the generic framework of upgrading classical reductions to non-local reductions, introduced in [AKL23], can be leveraged to extend the above result to large values of q.

In the case when $\mathcal{D}_{\mathsf{bit}}$ is not a uniform distribution, showing the hardness of the non-locally solving the above problem seems much harder.

Specifically, we are interested in the following setting: $\mathcal{D}_{\mathsf{bit}}$ is a distribution on $\{0,1\} \times \{0,1\}$, where (b, b) is sampled with probability $\frac{1}{2}$, for $b \in \{0,1\}$. In this case, we simply refer to the above problem as $(\mathcal{D}_{\mathcal{X}}, \mathcal{D}_{\mathsf{Ch}})$-simultIP problem.

Conjectures. We state the following conjectures. We are interested in the following distributions:

- We define $\mathcal{D}_{\mathsf{Ch}}^{\mathsf{ind}}$ as follows: it samples $\left((\mathbf{u}^{\mathcal{B}}, m^{\mathcal{B}}), (\mathbf{u}^{\mathcal{C}}, m^{\mathcal{C}})\right)$, where $\mathbf{u}^{\mathcal{B}} \xleftarrow{\$} \mathbb{Z}_q^n, \mathbf{u}^{\mathcal{C}} \xleftarrow{\$} \mathbb{Z}_q^n, m^{\mathcal{B}} \xleftarrow{\$} \mathbb{Z}_q, m^{\mathcal{C}} \xleftarrow{\$} \mathbb{Z}_q$. We define $\mathcal{D}_{\mathsf{Ch}}^{\mathsf{ind}}$ as follows: it samples $((\mathbf{u}, m), (\mathbf{u}, m))$, where $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^n, m \xleftarrow{\$} \mathbb{Z}_q$.
- Similarly, we define $\mathcal{D}_{\mathcal{X}}^{\mathsf{ind}}$ as follows: it samples $\left(\mathbf{x}^{\mathcal{B}}, \mathbf{x}^{\mathcal{C}}\right)$, where $\mathbf{x}^{\mathcal{B}} \xleftarrow{\$} \mathbb{Z}_q^n, \mathbf{x}^{\mathcal{C}} \xleftarrow{\$} \mathbb{Z}_q^n$. We define $\mathcal{D}_{\mathcal{X}}^{\mathsf{id}}$ as follows: it samples $(\mathbf{x}, \mathbf{x})$, where $\mathbf{x} \xleftarrow{\$} \mathbb{Z}_q^n$.

Conjecture 13 ($(\mathcal{D}_{\mathcal{X}}^{\mathsf{id}}, \mathcal{D}_{\mathsf{Ch}}^{\mathsf{id}})$-simultIP Conjecture)**.** *Consider a set of bipartite states $\rho = \{\rho_{\mathbf{x}}\}_{\mathbf{x} \in \mathbb{Z}_q^n}$ satisfying the following property: for any QPT adversaries $\mathcal{B}, \mathcal{C}$,*

$$\mathsf{Pr}\left[(\mathbf{x}, \mathbf{x}) \leftarrow (\mathcal{B} \otimes \mathcal{C})(\rho_{\mathbf{x}}) \ : \ (\mathbf{x}, \mathbf{x}) \leftarrow \mathcal{D}_{\mathcal{X}}^{\mathsf{id}}\right] \leq \nu(n)$$

for some negligible function $\nu(\lambda)$.

Any QPT non-local solver for the $\left(\mathcal{D}_{\mathcal{X}}^{\mathsf{id}}, \mathcal{D}_{\mathsf{Ch}}^{\mathsf{id}}\right)$-simultIP *problem succeeds with probability at most* $\frac{1}{2} + \varepsilon(n)$*, where* ε *is a negligible function.*

Conjecture 14 (($\mathcal{D}_{\mathcal{X}}^{\mathsf{ind}}, \mathcal{D}_{\mathsf{Ch}}^{\mathsf{ind}}$)-simultIP Conjecture)**.** *Consider a set of bipartite states* $\rho = \{\rho_{\mathbf{x}^{\mathcal{B}},\mathbf{x}^{\mathcal{C}}}\}_{\mathbf{x}^{\mathcal{B}},\mathbf{x}^{\mathcal{C}} \in \mathbb{Z}_q^n}$ *satisfying the following property: for any QPT adversaries* $\mathcal{B}, \mathcal{C}$*,*

$$\Pr\left[\left(\mathbf{x}^{\mathcal{B}}, \mathbf{x}^{\mathcal{C}}\right) \leftarrow (\mathcal{B} \otimes \mathcal{C})\left(\rho_{\mathbf{x}^{\mathcal{B}},\mathbf{x}^{\mathcal{C}}}\right) \ : \ \left(\mathbf{x}^{\mathcal{B}}, \mathbf{x}^{\mathcal{C}}\right) \leftarrow \mathcal{D}_{\mathcal{X}}^{\mathsf{ind}}\right] \leq \nu(n)$$

for some negligible function $\nu(\lambda)$.

Any QPT non-local solver for the $\mathcal{D}_{\mathsf{Ch}}^{\mathsf{id}}$-simultIP *problem succeeds with probability at most* $\frac{1}{2}+\varepsilon(n)$*, where* ε *is a negligible function.*

5 Construction of Unclonable Puncturable Obfuscation

In this section, we construct unclonable puncturable obfuscation for all efficiently computable generalized puncturable keyed circuit classes, with respect to $\mathcal{U}$ and $\mathsf{Id}_{\mathcal{U}}$ challenge distribution (see Section 3.1.1). Henceforth, we assume that any keyed circuit class we consider will consist of circuits that are efficiently computable.

We present the construction in three steps.

1. In the first step (Section 5.1), we construct a single decryptor encryption scheme based on the CLLZ scheme [CLLZ21] (see Figure 4) and show that it satisfies $\mathcal{D}_{\mathsf{ind\text{-}msg}}$-indistinguishability from random anti-piracy (and $\mathcal{D}_{\mathsf{ind\text{-}msg}}$-indistinguishability from random anti-piracy respectively) (see Appendix A.2), based on the conjectures, Conjectures 13 and 14.

2. In the second step (Section 5.2), we define a variant of the security definition considered in [CLLZ21] with respect to two different challenge distributions and prove that the copy-protection construction for PRFs in [CLLZ21] (see Figure 8) satisfies this security notion, based on the indistinguishability from random anti-piracy guarantees of the single decryptor encryption scheme considered in the first step.

3. In the third step (Section 5.3), we show how to transform the copy-protection scheme obtained from the first step into UPO for a keyed circuit class with respect to the $\mathcal{U}$ and $\mathsf{Id}_{\mathcal{U}}$ challenge distribution.

5.1 A New Public-Key Single-Decryptor Encryption Scheme

The first step is to construct a single-decryptor encryption of the suitable form. While single-decryptor encryption has been studied in prior works [GZ20, CLLZ21], we require indistinguishability from random anti-piracy, see Appendix A.2, which has not been considered in prior works.

Our construction is based on the single decryptor encryption scheme in [CLLZ21, Section 6.3]. Hence, we first recall the CLLZ single decryptor encryption scheme, given in Figure 4.

Next in Section 5.1.1, we define a family of single decryptor encryption schemes based on the CLLZ single decryptor encryption, called CLLZ *post-processing* schemes, and then in Section 5.1.2, we give a construction of CLLZ *post-processing* single decryptor encryption scheme (Figure 6). Unfortunately, we are able to prove the required security guarantees of this construction only assuming conjectures that state the simultaneous versions of the Goldreich-Levin guarantees, see Conjectures 13 and 14, given in Section 4.

Assumes: post-quantum indistinguishability obfuscation iO.

$\mathsf{Gen}(1^\lambda)$:

1. Sample ℓ_0 uniformly random subspaces $\{A_i\}_{i\in[\ell_0]}$ and for each $i \in [\ell_0]$, sample s_i, s'_i.
2. Compute $\{R_i^0, R_i^1\}_{i\in\ell_0}$, where for every $i \in [\ell_0]$, $R_i^0 \leftarrow \mathsf{iO}(A_i + s_i)$ and $R_i^1 \leftarrow \mathsf{iO}(A_i^\perp + s'_i)$ are the membership oracles.
3. Output $\mathsf{sk} = \{\{A_{i s_i, s'_i}\}_i\}$ and $\mathsf{pk} = \{R_i^0, R_i^1\}_{i\in\ell_0}$

$\mathsf{QKeyGen}(\mathsf{sk})$:

1. Interprete sk as $\{\{A_{i s_i, s'_i}\}_i\}$.
2. Output $\rho_s k = \{\{|A_{i s_i, s'_i}\rangle\}_i\}$.

$\mathsf{Enc}(\mathsf{pk}, m)$:

1. Interprete $\mathsf{pk} = \{R_i^0, R_i^1\}_{i\in\ell_0}$.
2. Sample $r \xleftarrow{\$} \{0,1\}^n$.
3. Generate $\tilde{Q} \leftarrow \mathsf{iO}(Q_{m,r})$ where $Q_{m,r}$ has $\{R_i^0, R_i^1\}_{i\in\ell_0}$ hardcoded inside, and on input $v_1, \ldots, v_{\ell_0} \in \{0,1\}^{n\ell_0}$, checks if $R_i^{r_i}(v_i) = 1$ for every $i \in [\ell_0]$ and if the check succeeds, outputs m, otherwise output $\perp$.
4. Output $\mathsf{ct} = (r, \tilde{Q})$

$\mathsf{Dec}(\rho_{\mathsf{sk}}, \mathsf{ct})$

1. Interprete $\mathsf{ct} = (r, \tilde{Q})$.
2. For every $i \in [\ell_0]$, if $r_i = 1$ apply $H^{\otimes n}$ on $|A_{i s_i, s'_i}\rangle$. Let the resulting state be $|\psi_x\rangle$.
3. Run the circuit $\tilde{Q}$ in superposition on the state $|\psi_x\rangle$ and measure the output register and output the measurement result m.

Figure 4: The CLLZ single decryptor encryption scheme, see [CLLZ21, Construction 1].

5.1.1 Definition of a CLLZ post-processing single decryptor encryption scheme

We call a single decryptor encryption scheme $(\mathsf{Gen}, \mathsf{QKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ CLLZ *post-processing* if there exists polynomial time classical deterministic algorithms $(\mathsf{EncPostProcess}, \mathsf{DecPostProcess})$, such that EncPostProcess has input length $2q(\lambda)$ and output length $s(\lambda)$, and DecPostProcess has input length $q(\lambda) + s(\lambda)$ and output length $q(\lambda)$, where $q(\lambda)$ is the length of the messages for the CLLZ single decryptor encryption scheme (see Figure 4) and $s(\lambda) \in \mathrm{poly}(\lambda)$, such that it is of the form described in Figure 5. For correctness of, a CLLZ *post-processing* single decryptor encryption scheme we require that for every string $r, m \in \{0,1\}^q$,

$$c' \leftarrow \mathsf{EncPostProcess}(m, r), m' \leftarrow \mathsf{DecPostProcess}(c', r) \implies m = m'. \tag{1}$$

Note that if the above condition is satisfied then it holds that for every $\delta \in [0,1]$, δ-correctness of the CLLZ single decryptor encryption implies δ-correctness of the CLLZ *post-processing* single decryptor encryption (see Figure 5). Note that if the above condition is satisfied then it holds that for every $\delta \in [0,1]$, δ-correctness of the CLLZ single decryptor encryption implies δ-correctness of the CLLZ *post-processing* single decryptor encryption (see Figure 5)

Assumes: CLLZ single decryptor encryption scheme given in Figure 4.

$\mathsf{Gen}(1^\lambda)$: Same as $\mathsf{CLLZ.Gen}(1^\lambda)$.

$\mathsf{QKeyGen}(\mathsf{sk})$: Same as $\mathsf{CLLZ.QKeyGen}(\mathsf{sk})$.

$\mathsf{Enc}(\mathsf{pk}, m)$:

1. Sample $r \xleftarrow{\$} \{0,1\}^q$.
2. Generate $c \leftarrow \mathsf{EncPostProcess}(m, r)$ and generate $c' \leftarrow \mathsf{CLLZ.Enc}(\mathsf{pk}, r)$[6].
3. Output $\mathsf{ct} = (c, c')$.

$\mathsf{Dec}(\rho_{\mathsf{sk}}, \mathsf{ct})$

1. Interprete $\mathsf{ct} = (c, c')$.
2. Generate $r \leftarrow \mathsf{CLLZ.Dec}(\rho_{\mathsf{sk}}, c')$.
3. Output $m \leftarrow \mathsf{DecPostProcess}(c, r)$.

Figure 5: Definition of a CLLZ *post-processing* single decryptor encryption scheme.

5.1.2 Construction of a CLLZ post-processing single decryptor encryption scheme

We next consider the following CLLZ *post-processing* scheme where

$\mathsf{EncPostProcess}(m, r)$:

1. Sample $u \xleftarrow{\$} \{0,1\}^q$.
2. Output $u, m \oplus \langle u, r \rangle$, where the innerproduct is the product over the field $\mathbb{F}_Q$ where Q is the smallest prime number greater than 2^q,.

$\mathsf{DecPostProcess}(c, r)$:

1. Interprete c as u, z.
2. Output $z \oplus \langle u, r \rangle$.

Figure 6: Construction of a CLLZ *post-processing* single decryptor encryption scheme.

Note that $\mathsf{EncPostProcess}, \mathsf{DecPostProcess}$ in Figure 6 satisfies Equation (1), and hence if the CLLZ single decryptor encryption scheme (depicted in Figure 4) satisfies δ-correctness so does the single decryptor encryption scheme in Figure 6.

Next we prove that the single decryptor encryption scheme in Figure 6 satisfies $\mathcal{D}_{\mathsf{ind\text{-}msg}}$-indistinguishability from random anti-piracy and $\mathcal{D}_{\mathsf{identical\text{-}cipher}}$-indistinguishability from random anti-piracy by exploiting the corresponding simultaneous Goldreich-Levin conjectures (see Conjectures 13 and 14).

[6] We would like to note that the obfuscated circuit may be padded more than what is required in the CLLZ single decryptor encryption scheme, for the security proofs of the CLLZ *post-processing* single decryptor encryption.

Theorem 15. *Assuming Conjecture 14, the existence of post-quantum sub-exponentially secure* iO *and one-way functions, and the quantum hardness of Learning-with-errors problem (LWE), the* CLLZ *post-processing* single decryptor encryption *as defined in Figure 5 given in Figure 6 satisfies* $\mathcal{D}_{\mathsf{ind\text{-}msg}}$*-indistinguishability from random anti-piracy (see Appendix A.2).*

Proof. Let $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ be an adversary against the single decryptor encryption scheme CLLZ Post-Process given in Figure 4 in the $\mathcal{D}_{\mathsf{ind\text{-}msg}}$-indistinguishability from random anti-piracy experiment (see Game 34). We will do a sequence of hybrids; the changes would be marked in blue.
Hybrid$_0$: Same as $\mathsf{Ind\text{-}random.SDE.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}_{\mathsf{ind\text{-}msg}}}(1^\lambda)$ (see Game 34) where $\mathcal{D}_{\mathsf{ind\text{-}msg}}$ is the challenge distribution defined in Appendix A.2 for the single-decryptor encryption scheme, CLLZ Post-Process in Figure 6.

1. Ch samples $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $\rho_k \leftarrow \mathsf{QKeyGen}(k)$ and sends ρ_k, pk to $\mathcal{A}$.
2. $\mathcal{A}(\rho_k, \mathsf{pk})$ outputs $\sigma_{\mathcal{B},\mathcal{C}}$.
3. Ch samples $b \xleftarrow{\$} \{0,1\}$.
4. Ch computes $\mathsf{ct}_b^{\mathcal{B}}$ as follows:
   (a) Sample $r^{\mathcal{B}} \xleftarrow{\$} \{0,1\}^q$, and compute $c'^{\mathcal{B}} \leftarrow \mathsf{CLLZ.Enc}(\mathsf{pk}, r^{\mathcal{B}})$.
   (b) Sample $u^{\mathcal{B}} \xleftarrow{\$} \{0,1\}^q$ and compute $c_b^{\mathcal{B}} = (u^{\mathcal{B}}, \langle u^{\mathcal{B}}, r^{\mathcal{B}}\rangle)$ if $b = 0$, else sample $m^{\mathcal{B}} \xleftarrow{\$} \{0,1\}^q$ and compute $c_b^{\mathcal{B}} = (u^{\mathcal{B}}, m^{\mathcal{B}} \oplus \langle u^{\mathcal{B}}, r^{\mathcal{B}}\rangle)$ if $b = 1$.
   (c) Set $\mathsf{ct}_b^{\mathcal{B}} = (c_b^{\mathcal{B}}, c'^{\mathcal{B}})$.
5. Ch computes $\mathsf{ct}_b^{\mathcal{C}}$ as follows:
   (a) Sample $r^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^q$, and compute $c'^{\mathcal{C}} \leftarrow \mathsf{CLLZ.Enc}(\mathsf{pk}, r^{\mathcal{C}})$.
   (b) Sample $u^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^q$ and compute $c_b^{\mathcal{C}} = (u^{\mathcal{C}}, \langle u^{\mathcal{C}}, r^{\mathcal{C}}\rangle)$ if $b = 0$, else sample $m^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^q$ and compute $c_b^{\mathcal{C}} = (u^{\mathcal{C}}, m^{\mathcal{C}} \oplus \langle u^{\mathcal{C}}, r^{\mathcal{C}}\rangle)$ if $b = 1$.
   (c) Set $\mathsf{ct}_b^{\mathcal{C}} = (c_b^{\mathcal{C}}, c'^{\mathcal{C}})$.
6. Apply $(\mathcal{B}(\mathsf{ct}_b^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}_b^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_{\mathbf{B}}, b_{\mathbf{C}})$.
7. Output 1 if $b_{\mathbf{B}} = b_{\mathbf{C}} = b$.

Hybrid$_1$:

1. Ch samples $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $\rho_k \leftarrow \mathsf{QKeyGen}(k)$ and sends ρ_k, pk to $\mathcal{A}$.
2. $\mathcal{A}(\rho_k, \mathsf{pk})$ outputs $\sigma_{\mathcal{B},\mathcal{C}}$.
3. Ch samples $b \xleftarrow{\$} \{0,1\}$.
4. Ch computes $\mathsf{ct}_b^{\mathcal{B}}$ as follows:
   (a) Sample $r^{\mathcal{B}} \xleftarrow{\$} \{0,1\}^q$, and compute $c'^{\mathcal{B}} \leftarrow \mathsf{CLLZ.Enc}(\mathsf{pk}, r^{\mathcal{B}})$.

(b) Sample $u^{\mathcal{B}} \xleftarrow{\$} \{0,1\}^q$ and compute $c_b^{\mathcal{B}} = (u^{\mathcal{B}}, \langle u^{\mathcal{B}}, r^{\mathcal{B}} \rangle)$ if $b = 0$, else ~~sample $m^{\mathcal{B}} \xleftarrow{\$} \{0,1\}^q$ and compute $c_b^{\mathcal{B}} = (u^{\mathcal{B}}, m^{\mathcal{B}} \oplus \langle u^{\mathcal{B}}, r^{\mathcal{B}} \rangle)$ if $b = 1$~~ compute $c_b^{\mathcal{B}} = (u^{\mathcal{B}}, m^{\mathcal{B}})$ if $b = 1$.

(c) Set $\mathsf{ct}_b^{\mathcal{B}} = (c_b^{\mathcal{B}}, c'^{\mathcal{B}})$.

5. Ch computes $\mathsf{ct}_b^{\mathcal{C}}$ as follows:

   (a) Sample $r^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^q$, and compute $c'^{\mathcal{C}} \leftarrow \mathsf{CLLZ.Enc}(\mathsf{pk}, r^{\mathcal{C}})$.

   (b) Sample $u^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^q$ and compute $c_b^{\mathcal{C}} = (u^{\mathcal{C}}, \langle u^{\mathcal{C}}, r^{\mathcal{C}} \rangle)$ if $b = 0$, else ~~sample $m^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^q$ and compute $c_b^{\mathcal{C}} = (u^{\mathcal{C}}, m^{\mathcal{C}} \oplus \langle u^{\mathcal{C}}, r^{\mathcal{C}} \rangle)$ if $b = 1$~~ compute $c_b^{\mathcal{C}} = (u^{\mathcal{C}}, m^{\mathcal{C}})$ if $b = 1$.

   (c) Set $\mathsf{ct}_b^{\mathcal{C}} = (c_b^{\mathcal{C}}, c'^{\mathcal{C}})$.

6. Apply $(\mathcal{B}(\mathsf{ct}_b^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}_b^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_{\mathbf{B}}, b_{\mathbf{C}})$.

7. Output 1 if $b_{\mathbf{B}} = b_{\mathbf{C}} = b$.

The indistinguishability holds since the overall distribution of $\mathsf{ct}_b^{\mathcal{B}}$ and $\mathsf{ct}_b^{\mathcal{C}}$ did not change across hybrids Hybrid_0 and Hybrid_1.

Consider the following independent search experiment against a pair of (uniform) efficient adversaries $\mathcal{B}', \mathcal{C}'$

1. Ch samples $r^{\mathcal{B}}, r^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^q$

2. Ch computes $\sigma_{\mathcal{B},\mathcal{C}}$ as follows:

   (a) Sample $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and prepares $\rho_k \leftarrow \mathsf{QKeyGen}(k)$.

   (b) Run $\mathcal{A}(\rho_k, \mathsf{pk})$ to get $\sigma_{\mathcal{B},\mathcal{C}}$.

3. Ch computes $c'^{\mathcal{B}} \leftarrow \mathsf{CLLZ.Enc}(\mathsf{pk}, r^{\mathcal{B}})$, and computes $c'^{\mathcal{C}} \leftarrow \mathsf{CLLZ.Enc}(\mathsf{pk}, r^{\mathcal{C}})$.

4. Ch constructs the bipartite auxilliary state $\tau_{B,C}^{r^{\mathcal{B}}, r^{\mathcal{C}}} = c'^{\mathcal{B}}, \sigma_{\mathcal{B},\mathcal{C}}, c'^{\mathcal{C}}$, i.e., the $c'^{\mathcal{B}}, \sigma_{\mathcal{B}}$ and $c'^{\mathcal{C}}, \sigma_{\mathcal{C}}$ are the two partitions.

5. Ch sends the respective registers of $\tau_{B,C}^{r^{\mathcal{B}}, r^{\mathcal{C}}}$ to $\mathcal{B}'$ and $\mathcal{C}'$, and gets back the responses $r'^{\mathcal{B}}$ and $r'^{\mathcal{C}}$ respectively.

6. Ouptput 1 if $r'^{\mathcal{B}} = r^{\mathcal{B}}$, and $r'^{\mathcal{C}} = r^{\mathcal{C}}$.

Clearly, the winning probability of $(\mathcal{B}', \mathcal{C}')$ in the above game is the same as the winning probability of $(\mathcal{A}, \mathcal{B}', \mathcal{C}')$ in the independent search anti-piracy (see Appendix A.2) of the CLLZ single decryptor encryption scheme given in Figure 4. It was shown in [CLLZ21, Theorem 6.15] that the CLLZ single decryptor encryption satisfies independent search anti-piracy assuming the security guarantess of post-quantum sub-exponentially secure iO and one-way functions, and quantum hardness of Learning-with-errors problem (LWE). Hence, under the security guarantees of the above assumptions, there exists a negligible function $\epsilon'()$ such that the winning probability of $(\mathcal{B}', \mathcal{C}')$ in the above game is $\epsilon'(\lambda)$. Therefore assuming Conjecture 14, there exists a negligible function $\epsilon()$ such that the winning probability of $(\mathcal{B}, \mathcal{C})$ in the following indistinguishability game is at most $\frac{1}{2} + \epsilon(\lambda)$

1. Ch samples $r^{\mathcal{B}}, r^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^q$

2. Ch computes $\sigma_{\mathcal{B},\mathcal{C}}$ as follows:

   (a) Sample $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and prepares $\rho_k \leftarrow \mathsf{QKeyGen}(k)$.

   (b) Run $\mathcal{A}(\rho_k, \mathsf{pk})$ to get $\sigma_{\mathcal{B},\mathcal{C}}$.

3. Ch computes $c'^{\mathcal{B}} \leftarrow \mathsf{CLLZ.Enc}(\mathsf{pk}, r^{\mathcal{B}})$, and computes $c'^{\mathcal{C}} \leftarrow \mathsf{CLLZ.Enc}(\mathsf{pk}, r^{\mathcal{C}})$.

4. Ch constructs the bipartite auxilliary state $\tau_{B,C}^{r^{\mathcal{B}},r^{\mathcal{C}}} = c'^{\mathcal{B}}, \sigma_{\mathcal{B},\mathcal{C}}, c'^{\mathcal{C}}$, i.e., the $c'^{\mathcal{B}}, \sigma_{\mathcal{B}}$ and $c'^{\mathcal{C}}, \sigma_{\mathcal{C}}$ are the two partitions.

5. Ch samples $b \xleftarrow{\$} \{0,1\}$.

6. Ch samples $u^{\mathcal{B}} \xleftarrow{\$} \{0,1\}^q$ and compute $c_b^{\mathcal{B}} = (u^{\mathcal{B}}, \langle u^{\mathcal{B}}, r^{\mathcal{B}} \rangle)$ if $b = 0$, else computes $c_b^{\mathcal{B}} = (u^{\mathcal{B}}, m^{\mathcal{B}})$ if $b = 1$.

7. Similarly, Ch samples $u^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^q$ and computes $c_b^{\mathcal{C}} = (u^{\mathcal{C}}, \langle u^{\mathcal{C}}, r^{\mathcal{C}} \rangle)$ if $b = 0$, else computes $c_b^{\mathcal{C}} = (u^{\mathcal{C}}, m^{\mathcal{C}})$ if $b = 1$.

8. Ch sends $c_b^{\mathcal{B}}$ and $c_b^{\mathcal{C}}$ along with the respective registers of $\tau_{B,C}^{r^{\mathcal{B}},r^{\mathcal{C}}}$ to $\mathcal{B}'$ and $\mathcal{C}'$ respectively, and gets back the responses $b^{\mathcal{B}}$ and $b^{\mathcal{C}}$ respectively.

9. Output 1 if $b_{\mathbf{B}} = b_{\mathbf{C}} = b$.

However, note that the view of the adversaries $\mathcal{B}$ and $\mathcal{C}$ in the indistinguishability game above is the same as the view in Hybrid_3. Therefore, the winning probability of $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ in Hybrid_1 is at most $\frac{1}{2} + \epsilon(\lambda)$. This completes the proof of the theorem. □

Theorem 16. *Assuming Conjecture 13, the existence of post-quantum sub-exponentially secure* iO *and one-way functions, and quantum hardness of Learning-with-errors problem (LWE), the* CLLZ *post-processing* single decryptor encryption *(as defined in Figure 5) given in Figure 6 satisfies $\mathcal{D}_{\mathsf{identical\text{-}cipher}}$-indistinguishability from random anti-piracy (see Appendix A.2).*

Proof. The proof directly follows by combining Lemmas 17 and 18. □

Lemma 17. *Assuming Conjecture 13, the* CLLZ *post-processing* single decryptor encryption *as defined in Figure 5 given in Figure 6 satisfies $\mathcal{D}_{\mathsf{identical\text{-}cipher}}$-indistinguishability from random anti-piracy, if* CLLZ single decryptor encryption *(see Figure 4) satisfies $\mathsf{Id}_{\mathcal{U}}$-search anti-piracy (see Appendix A.2).*

Proof. Let $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ be an adversary against the single decryptor encryption scheme CLLZ Post-Process given in Figure 4 in the $\mathcal{D}_{\mathsf{identical\text{-}cipher}}$-indistinguishability from random anti-piracy experiment. We will do a sequence of hybrids; the changes will be marked in blue.
$\underline{\mathsf{Hybrid}_0}$: Same as $\mathsf{Ind\text{-}random.SDE.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}_{\mathsf{identical\text{-}cipher}}}(1^\lambda)$ (see Game 34) where $\mathcal{D}_{\mathsf{identical\text{-}cipher}}$ is the challenge distribution defined in Appendix A.2 for the single-decryptor encryption scheme, CLLZ Post-Process in Figure 6.

1. Ch samples $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $\rho_k \leftarrow \mathsf{QKeyGen}(k)$ and sends ρ_k, pk to $\mathcal{A}$.
2. $\mathcal{A}(\rho_k, \mathsf{pk})$ outputs $\sigma_{\mathcal{B},\mathcal{C}}$.
3. Ch samples $b \xleftarrow{\$} \{0,1\}$.
4. Ch computes ct_b as follows:
   (a) Sample $r \xleftarrow{\$} \{0,1\}^q$, and compute $c' \leftarrow \mathsf{CLLZ.Enc}(\mathsf{pk}, r)$.
   (b) Sample $u \xleftarrow{\$} \{0,1\}^q$ and compute $c_b = (u, \langle u, r\rangle)$ if $b = 0$, else sample $m \xleftarrow{\$} \{0,1\}^q$ and compute $c_b = (u, m \oplus \langle u, r\rangle)$ if $b = 1$.
   (c) Set $\mathsf{ct}_b = (c_b, c')$.
5. Apply $(\mathcal{B}(\mathsf{ct}_b, \cdot) \otimes \mathcal{C}(\mathsf{ct}_b, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_{\mathbf{B}}, b_{\mathbf{C}})$.
6. Output 1 if $b_{\mathbf{B}} = b_{\mathbf{C}} = b$.

$\underline{\mathsf{Hybrid}_1}$:

1. Ch samples $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $\rho_k \leftarrow \mathsf{QKeyGen}(k)$ and sends ρ_k, pk to $\mathcal{A}$.
2. $\mathcal{A}(\rho_k, \mathsf{pk})$ outputs $\sigma_{\mathcal{B},\mathcal{C}}$.
3. Ch samples $b \xleftarrow{\$} \{0,1\}$.
4. Ch computes ct_b as follows:
   (a) Sample $r \xleftarrow{\$} \{0,1\}^q$, and compute $c' \leftarrow \mathsf{CLLZ.Enc}(\mathsf{pk}, r)$.
   (b) Sample $u \xleftarrow{\$} \{0,1\}^q$ and compute $c_b = (u, \langle u, r\rangle)$ if $b = 0$, else ~~sample $m \xleftarrow{\$} \{0,1\}^q$ and compute $c_b = (u, m \oplus \langle u, r\rangle)$ if $b = 1$~~ compute $c_b = (u, m)$ if $b = 1$.
   (c) Set $\mathsf{ct}_b = (c_b, c')$.
5. Apply $(\mathcal{B}(\mathsf{ct}_b, \cdot) \otimes \mathcal{C}(\mathsf{ct}_b, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_{\mathbf{B}}, b_{\mathbf{C}})$.
6. Output 1 if $b_{\mathbf{B}} = b_{\mathbf{C}} = b$.

The indistinguishability holds since the overall distribution of ct_b did not change across hybrids Hybrid_0 and Hybrid_1.

Consider the following search experiment against a pair of (uniform) efficient adversaries $\mathcal{B}', \mathcal{C}'$

1. Ch samples $r \xleftarrow{\$} \{0,1\}^q$.
2. Ch computes $\sigma_{\mathcal{B},\mathcal{C}}$ as follows:
   (a) Sample $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and prepares $\rho_k \leftarrow \mathsf{QKeyGen}(k)$.
   (b) Run $\mathcal{A}(\rho_k, \mathsf{pk})$ to get $\sigma_{\mathcal{B},\mathcal{C}}$.
3. Ch computes $c' \leftarrow \mathsf{CLLZ.Enc}(\mathsf{pk}, r)$.

4. Ch constructs the bipartite auxilliary state $\tau^r_{B,C} = c'^{\mathcal{B}}, \sigma_{\mathcal{B},\mathcal{C}}, c'^{\mathcal{C}}$, i.e., the $c'^{\mathcal{B}}, \sigma_{\mathcal{B}}$ and $c'^{\mathcal{C}}, \sigma_{\mathcal{C}}$ are the two partitions, where $c'^{\mathcal{B}} = c'^{\mathcal{C}} = c'$.

5. Ch sends the respective registers of $\tau^r_{B,C}$ to $\mathcal{B}'$ and $\mathcal{C}'$, and gets back the responses $r'^{\mathcal{B}}$ and $r'^{\mathcal{C}}$ respectively.

6. Ouptput 1 if $r'^{\mathcal{B}} = r'^{\mathcal{C}} = r$.

Clearly, the winning probability of $(\mathcal{B}', \mathcal{C}')$ in the above game is the same as the winning probability of $(\mathcal{A}, \mathcal{B}', \mathcal{C}')$ in the $\mathsf{Id}_{\mathcal{U}}$-search anti-piracy (see Appendix A.2) of the CLLZ single decryptor encryption scheme given in Figure 4. Assuming the CLLZ single decryptor encryption satisfies $\mathsf{Id}_{\mathcal{U}}$-search anti-piracy (see Appendix A.2), there exists a negligible function $\epsilon'()$ such that the winning probability of $(\mathcal{B}', \mathcal{C}')$ in the above game is $\epsilon'(\lambda)$. Therefore by Conjecture 14, there exists a negligible function $\epsilon()$ such that the winning probability of $(\mathcal{B}, \mathcal{C})$ in the following indistinguishability game is at most $\frac{1}{2} + \epsilon(\lambda)$

1. Ch samples $r \xleftarrow{\$} \{0,1\}^q$.

2. Ch computes $\sigma_{\mathcal{B},\mathcal{C}}$ as follows:

   (a) Sample $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and prepares $\rho_k \leftarrow \mathsf{QKeyGen}(k)$.

   (b) Run $\mathcal{A}(\rho_k, \mathsf{pk})$ to get $\sigma_{\mathcal{B},\mathcal{C}}$.

3. Ch computes $c' \leftarrow \mathsf{CLLZ.Enc}(\mathsf{pk}, r)$.

4. Ch constructs the bipartite auxilliary state $\tau^r_{B,C} = c'^{\mathcal{B}}, \sigma_{\mathcal{B},\mathcal{C}}, c'^{\mathcal{C}}$, i.e., the $c'^{\mathcal{B}}, \sigma_{\mathcal{B}}$ and $c'^{\mathcal{C}}, \sigma_{\mathcal{C}}$ are the two partitions, where $c'^{\mathcal{B}} = c'^{\mathcal{C}} = c'$.

5. Ch samples $b \xleftarrow{\$} \{0,1\}$.

6. Ch samples $u \xleftarrow{\$} \{0,1\}^q$ and compute $c_b = (u, \langle u, r\rangle)$ if $b = 0$, else computes $c_b = (u, m)$ if $b = 1$.

7. Ch sends $c_b^{\mathcal{B}}$ and $c_b^{\mathcal{C}}$ along with the respective registers of $\tau^{r^{\mathcal{B}}, r^{\mathcal{C}}}_{B,C}$ to $\mathcal{B}'$ and $\mathcal{C}'$ respectively, where $c_b^{\mathcal{B}} = c_b^{\mathcal{C}} = c_b$ and gets back the responses $b^{\mathcal{B}}$ and $b^{\mathcal{C}}$ respectively.

8. Output 1 if $b_{\mathbf{B}} = b_{\mathbf{C}} = b$.

However, note that the view of the adversaries $\mathcal{B}$ and $\mathcal{C}$ in the indistinguishability game above is the same as the view in Hybrid_3. Therefore, the winning probability of $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ in Hybrid_1 is at most $\frac{1}{2} + \epsilon(\lambda)$. This completes the proof of the lemma.

□

Lemma 18. *Assuimng post-quantum sub-exponentially secure* iO *and quantum hardness of Learning-with-errors problem (LWE), the* CLLZ single decryptor encryption *(see Figure 4) satisfies* $\mathsf{Id}_{\mathcal{U}}$*-search anti-piracy (see Appendix A.2).*

Proof. By [CLLZ21, Theorem 6.15], assuming the security of post-quantum sub-exponentially secure iO and one-way functions, and quantum hardness of Learning-with-errors problem (LWE), the CLLZ single decryptor encryption (see Figure 4) satisfies independent search anti-piracy. Since the trivial success probabilities of the $\mathcal{U}$-search anti-piracy and $\mathsf{Id}_{\mathcal{U}}$-search anti-piracy expeirments for single decryptor encryption are both negligible, by the lifting result in [AKL23, Theorem], we conclude that Lemma 18 holds. ☐

5.2 Copy-Protection for PRFs with Preponed Security

We first introduce the definition of *preponed security* in Section 5.2.1 and then we present the constructions of copy-protection in Section 5.2.2.

5.2.1 Definition

We introduce a new security notion for copy-protection called *preponed security*.

Consider a pseudorandom function family $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$, where $\mathcal{F}_\lambda = \{f_k : \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{\kappa(\lambda)} \; : \; k \in \{0,1\}^\lambda\}$. Moreover, f_k can be implemented using a polynomial-sized circuit, denoted by C_k.

Definition 19 (Preponed Security)**.** *A copy-protection scheme* $\mathsf{CP} = (\mathsf{CopyProtect}, \mathsf{Eval})$ *for* $\mathcal{F}$ *(Appendix A.1) satisfies* $\mathcal{D}_\mathcal{X}$*-preponed security if for any QPT* $(\mathcal{A}, \mathcal{B}, \mathcal{C})$*, there exists a negligible function* negl *such that:*

$$\Pr[\mathsf{PreponedExpt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{F},\mathcal{U}}\left(1^\lambda\right) = 1] \leq \frac{1}{2} + \mathsf{negl}.$$

where $\mathsf{PreponedExpt}$ *is defined in Figure 7.*

We consider two instantiations of $\mathcal{D}_\mathcal{X}$*:*

1. $\mathcal{U}$ *which is the product of uniformly random distribution on* $\{0,1\}^\ell$*, meaning* $x_1, x_2 \leftarrow \mathcal{U}(1^\lambda)$ *where* $x_1, x_2 \xleftarrow{\$} \{0,1\}^\ell$ *independently.*
2. $\mathsf{Id}_\mathcal{U}$ *which is the perfectly correlatd distribution on* $\{0,1\}^\ell$ *with uniform marginals, meaning* $x, x \leftarrow \mathsf{Id}_\mathcal{U}(1^\lambda)$ *where* $x \xleftarrow{\$} \{0,1\}^\ell$*.*

5.2.2 Construction

The CLLZ copy-protection scheme is given in Figure 8.

Construction of Copy-Protection.

Proposition 20. *Assuming the existence of post-quantum* iO*, and one-way functions, and if there exists a* CLLZ *post-processing* single decryptor encryption *scheme that satisfies* $\mathcal{D}_{\mathsf{ind\text{-}msg}}$*-indistinguishability from random anti-piracy, see Appendix A.2, then the* CLLZ *copy-protection construction in [CLLZ21, Section 7.3] (see Figure 8) satisfies* $\mathcal{U}$*-preponed security (Definition 19).*

Proposition 21. *Assuming the existence of post-quantum* iO*, and one-way functions, and if there exists a* CLLZ *post-processing* single decryptor encryption *scheme that satisfies* $\mathcal{D}_{\mathsf{identical\text{-}cipher}}$*-indistinguishability from random anti-piracy, see Appendix A.2, then the* CLLZ *copy-protection construction in [CLLZ21, Section 7.3] (see Figure 8) satisfies* $\mathsf{Id}_\mathcal{U}$*-preponed security (Definition 19).*

$\underline{\mathsf{PreponedExpt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathsf{CP},\mathcal{D}_\mathcal{X}}(1^\lambda)}$:

1. Ch samples $k \leftarrow \mathsf{KeyGen}(1^\lambda)$, then generates $\rho_{C_k} \leftarrow \mathsf{CopyProtect}(1^\lambda, C_k)$ and sends ρ_{f_k} to $\mathcal{A}$.
2. Ch samples $x^\mathcal{B}, x^\mathcal{C} \leftarrow \mathcal{D}_\mathcal{X}(1^\lambda)$, $b \xleftarrow{\$} \{0,1\}$. Let $y_1^\mathcal{B} = f(x^\mathcal{B}), y_1^\mathcal{C} = f(x^\mathcal{C})$, and $y_0^\mathcal{B} = y_1, y_0^\mathcal{C} = y_2$ where $y_1, y_2 \xleftarrow{\$} \{0,1\}^{\kappa(\lambda)}$. Ch gives $(y_b^\mathcal{B}, y_b^\mathcal{C})$ to Alice.
3. $\mathcal{A}(\rho_{C_k})$ outputs a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
4. Apply $(\mathcal{B}(x^\mathcal{B}, \cdot) \otimes \mathcal{C}(x^\mathcal{C}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_\mathbf{B}, b_\mathbf{C})$.
5. Output 1 if $b_\mathbf{B} = b_\mathbf{C} = b$.

Figure 7: Preponed security experiment for copy-protection of PRFs with respect to the distribution $\mathcal{D}_\mathcal{X}$.

Assumes: Punctrable and extractable PRF family $F_1 = (\mathsf{KeyGen}, \mathsf{Eval})$ (represented as $F_1(k,x) = \mathsf{PRF.Eval}(k,\cdot)$) and secondary PRF family F_2, F_3 with some special properties as noted in [CLLZ21]

$\mathsf{CopyProtect}(K_1)$:
1. Sample secondary keys K_2, K_3, and $\{\{|A_{i s_i, s'_i}\rangle\}_i\}$, and compute the coset state $\{\{|A_{i s_i, s'_i}\rangle\}_i\}$.
2. Compute $\tilde{P} \leftarrow \mathsf{iO}(P)$ where P is as given in Figure 11.
3. Output $\rho = (\tilde{P}, \{\{|A_{i s_i, s'_i}\rangle\}_i\})$.

$\mathsf{Eval}(\rho, x)$:
1. Interprete $\rho = (\tilde{P}, \{\{|A_{i s_i, s'_i}\rangle\}_i\})$.
2. Let $x = x_0 \| x_1 \| x_2$, where $x_0 = \ell_0$. For every $i \in [\ell_0]$, if $x_{0,i} = 1$ apply $H^{\otimes n}$ on $|A_{i s_i, s'_i}\rangle$. Let the resulting state be $|\psi_x\rangle$.
3. Run the circuit $\tilde{C}$ in superposition on the input registers (X, V) with the initial state $(x, |\psi_x\rangle)$ and then measure the output register to get an output y.

Figure 8: CLLZ copy-protection for PRFs.

Proof of Proposition 20. To prove the lemma, we adopt the proof of [CLLZ21, Theorem 7.12, Appendix F].

We will start with a series of hybrids. The changes are marked in blue.

$\underline{P}$:

Hardcoded keys $K_1, K_2, K_3, R_i^0, R_i^1$ for every $i \in [\ell_0]$ On input $x = x_0 \| x_1 \| x_2$ and vectors $v = v_1, \ldots v_{\ell_0}$.

1. If $F_3(K_3, x_1) \oplus x_2 = x_0 \| Q$ and $x_1 = F_2(K_2, x_0 \| Q)$:
   Hidden trigger mode: Treat Q as a classical circuit and output $Q(v)$.
2. Otherwise, check if the following holds: for all $i \in \ell_0$, $R^{x_{0,i}}(v_i) = 1$ (where $x_{0,i}$ is the i^{th} coordinate of x_0).
   Normal mode: If so, output $F_1(K_1, x)$ where $F_1() = \mathsf{PRF.Eval}()$ is the primary pseudo-random function family that is being copy-protected. Otherwise output $\perp$.

Figure 9: Circuit P in CLLZ copy-protection of PRF.

$\underline{\mathsf{Hybrid}_0}$: Same as $\mathsf{PreponedExpt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathsf{CP},\mathcal{D}_\mathcal{X}}(1^\lambda)$ (see Game 7) where $\mathcal{D} = \mathcal{U}$ (see the definition in Definition 19) for the CLLZ copy-protection scheme see Figure 8.

1. Ch samples $K_1 \leftarrow \mathsf{PRF.Gen}(1^\lambda)$ and generates $\rho = (\{|A_{i s_i, s'_i}\rangle\}_{i \in \ell_0}, \mathsf{iO}(P)) \leftarrow \mathsf{CLLZ.QKeyGen}(K_1)$, and sends ρ to $\mathcal{A}$. P has K_1, K_2, K_3 hardcoded in it where K_2, K_3 are the secondary keys.
2. Ch generates $x^\mathcal{B}, x^\mathcal{C} \xleftarrow{\$} \{0,1\}^n$, where $x^\mathcal{B} = x_0^\mathcal{B} \| x_1^\mathcal{B} \| x_2^\mathcal{B}, x^\mathcal{C} = x_0^\mathcal{C} \| x_1^\mathcal{C} \| x_2^\mathcal{C}$ and computes $y_0^\mathcal{B} \leftarrow \mathsf{PRF.Eval}(K_1, x^\mathcal{B})$ and $y_0^\mathcal{C} \leftarrow \mathsf{PRF.Eval}(K_1, x^\mathcal{C})$.
3. Ch also samples $y_1^\mathcal{B}, y_1^\mathcal{C} \xleftarrow{\$} \{0,1\}^m$.
4. Ch samples $b \xleftarrow{\$} \{0,1\}$, and sends $\mathcal{A}$, $(\rho, y_b^\mathcal{B}, y_b^\mathcal{C})$.
5. $\mathcal{A}$ on receiving $(\rho, y_b^\mathcal{B}, y_b^\mathcal{C})$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
6. Apply $(\mathcal{B}(x^\mathcal{B}, \cdot) \otimes \mathcal{C}(x^\mathcal{C}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^\mathcal{B}, b^\mathcal{C})$.
7. Output 1 if $b^\mathcal{B} = b^\mathcal{C} = b$, else 0.

$\underline{\mathsf{Hybrid}_1}$: We modify the sampling procedure of the challenge inputs $x^\mathcal{B}$ and $x^\mathcal{C}$.

1. Ch samples $K_1 \leftarrow \mathsf{PRF.Gen}(1^\lambda)$ and generates $\rho = (\{|A_{i s_i, s'_i}\rangle\}_{i \in \ell_0}, \mathsf{iO}(P)) \leftarrow \mathsf{CLLZ.QKeyGen}(K_1)$, and sends ρ to $\mathcal{A}$. P has K_1, K_2, K_3 hardcoded in it where K_2, K_3 are the secondary keys.
2. Ch generates $x^\mathcal{B}, x^\mathcal{C} \xleftarrow{\$} \{0,1\}^n$, where $x^\mathcal{B} = x_0^\mathcal{B} \| x_1^\mathcal{B} \| x_2^\mathcal{B}, x^\mathcal{C} = x_0^\mathcal{C} \| x_1^\mathcal{C} \| x_2^\mathcal{C}$ and computes $y_0^\mathcal{B} \leftarrow \mathsf{PRF.Eval}(K_1, x^\mathcal{B})$ and $y_0^\mathcal{C} \leftarrow \mathsf{PRF.Eval}(K_1, x^\mathcal{C})$.
3. Ch also computes $x_{\mathsf{trigger}}^\mathcal{B} \leftarrow \mathsf{Gen\text{-}Trigger}(x_0^\mathcal{B}, y_0^\mathcal{B}, K_2, K_3, \{A_{i s_i, s'_i}\}_{i \in \ell_0})$,
   and $x_{\mathsf{trigger}}^\mathcal{C} \leftarrow \mathsf{Gen\text{-}Trigger}(x_0^\mathcal{C}, y^\mathcal{C}, K_2, K_3, \{A_{i s_i, s'_i}\}_{i \in \ell_0})$.

4. Ch also samples $y_1^{\mathcal{B}}, y_1^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^m$.

5. Ch samples $b \xleftarrow{\$} \{0,1\}$ and sends $\mathcal{A}$ $(\rho, y_b^{\mathcal{B}}, y_b^{\mathcal{C}})$.

6. $\mathcal{A}$ on receiving $(\rho, y_b^{\mathcal{B}}, y_b^{\mathcal{C}})$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.

7. Apply $(\mathcal{B}((\ \sout{x^{\mathcal{B}}}\ x^{\mathcal{B}}_{\mathsf{trigger}}\ , \cdot) \otimes \mathcal{C}((\ \sout{x^{\mathcal{C}}}\ x^{\mathcal{C}}_{\mathsf{trigger}}\ , \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.

8. Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$, else 0.

Claim 22. *Assuming the security of* PRF*, hybrids* Hybrid_1 *and* Hybrid_2 *are computationally indistinguishable.*

Proof. Hybrid_1 is computationally indistinguishable from Hybrid_0 due to [CLLZ21, Lemma 7.17]. The same arguments via [CLLZ21, Lemma 7.17] were made in showing the indistinguishability between hybrids Hybrid_0 and Hybrid_1 in the proof of [CLLZ21, Theorem 7.12].

□

<u>Hybrid_2</u>: We modify the generation of the outputs $y_0^{\mathcal{B}}$ and $y_0^{\mathcal{C}}$.

1. Ch samples $K_1 \leftarrow \mathsf{PRF.Gen}(1^\lambda)$ and generates $\rho = (\{|A_{i s_i, s'_i}\rangle\}_{i \in \ell_0}, \mathsf{iO}(P)) \leftarrow \mathsf{CLLZ.QKeyGen}(K_1)$, and sends ρ to $\mathcal{A}$. P has K_1, K_2, K_3 hardcoded in it where K_2, K_3 are the secondary keys.

2. Ch generates $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$, where $x^{\mathcal{B}} = x_0^{\mathcal{B}} \| x_1^{\mathcal{B}} \| x_2^{\mathcal{B}}$, $x^{\mathcal{C}} = x_0^{\mathcal{C}} \| x_1^{\mathcal{C}} \| x_2^{\mathcal{C}}$ and ~~computes $y_0^{\mathcal{B}} \leftarrow \mathsf{PRF.Eval}(K_1, x^{\mathcal{B}})$ and $y_0^{\mathcal{C}} \leftarrow \mathsf{PRF.Eval}(K_1, x^{\mathcal{C}})$~~ samples $y_0^{\mathcal{B}}, y_0^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^m$.

3. Ch also computes $x^{\mathcal{B}}_{\mathsf{trigger}} \leftarrow \mathsf{Gen\text{-}Trigger}(x_0^{\mathcal{B}}, y_0^{\mathcal{B}}, K_2, K_3, \{A_{i s_i, s'_i}\}_{i \in \ell_0})$, and $x^{\mathcal{C}}_{\mathsf{trigger}} \leftarrow \mathsf{Gen\text{-}Trigger}(x_0^{\mathcal{C}}, y^{\mathcal{C}}, K_2, K_3, \{A_{i s_i, s'_i}\}_{i \in \ell_0})$.

4. Ch also samples $y_1^{\mathcal{B}}, y_1^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^m$.

5. Ch samples $b \xleftarrow{\$} \{0,1\}$ and sends $\mathcal{A}$ $(\rho, y_b^{\mathcal{B}}, y_b^{\mathcal{C}})$.

6. $\mathcal{A}$ on receiving $(\rho, y_b^{\mathcal{B}}, y_b^{\mathcal{C}})$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.

7. Apply $(\mathcal{B}(x^{\mathcal{B}}_{\mathsf{trigger}}, \cdot) \otimes \mathcal{C}(x^{\mathcal{C}}_{\mathsf{trigger}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.

8. Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$, else 0.

Hybrid_2 is statistically indistinguishable from Hybrid_1 due to the extractor properties of the primary PRF family. For more details, refer to the proof of see [CLLZ21, Theorem 7.12].

Claim 23. *Assuming the extractor properties of* PRF*, hybrids* Hybrid_2 *and* Hybrid_3 *are statistically indistinguishable.*

Proof. The proof is identical to the proof of indistinguishability of Hybrid_1 and Hybrid_2 in the proof of [CLLZ21, Theorem 7.12]. □

<u>Hybrid_3</u>: This hybrid is a reformulation of Hybrid_2 in terms of the CLLZ single decryptor encryption scheme, see fig. 4.

1. Ch samples $\{A_{is_i,s'_i}\}_{i\in\ell_0}$ and generates $\{|A_{is_i,s'_i}\rangle\}_{i\in\ell_0}$, and treats it as the quantum decryption key for the CLLZ single-decryptor encryption scheme (see fig. 4), where the secret key is $\{A_{is_i,s'_i}\}_{i\in\ell_0}$. Ch also generates $\mathsf{pk} = \{R_i^0, R_i^1\}_{i\in\ell_0}$, where for every $i \in [\ell_0]$, $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i^{\perp} + s'_i)$.
2. Ch ~~generates $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$, where $x^{\mathcal{B}} = x_0^{\mathcal{B}} \| x_1^{\mathcal{B}} \| x_2^{\mathcal{B}}, x^{\mathcal{C}} = x_0^{\mathcal{C}} \| x_1^{\mathcal{C}} \| x_2^{\mathcal{C}}$ and~~ samples $y_0^{\mathcal{B}}, y_0^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^m$.
3. ~~Ch also computes $x^{\mathcal{B}}_{\mathsf{trigger}} \leftarrow \mathsf{Gen\text{-}Trigger}(x_0^{\mathcal{B}}, y_0^{\mathcal{B}}, K_2, K_3, \{A_{is_i,s'_i}\}_{i\in\ell_0})$, and $x^{\mathcal{C}}_{\mathsf{trigger}} \leftarrow \mathsf{Gen\text{-}Trigger}(x_0^{\mathcal{C}}, y^{\mathcal{C}}, K_2, K_3, \{A_{is_i,s'_i}\}_{i\in\ell_0})$.~~
4. Ch also samples $y_1^{\mathcal{B}}, y_1^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^m$.
5. Ch samples $b \xleftarrow{\$} \{0,1\}$, and generates $x_0^{\mathcal{B}}, Q^{\mathcal{B}} \leftarrow \mathsf{CLLZ.Enc}(\mathsf{pk}, y_b^{\mathcal{B}})$ and $x_0^{\mathcal{C}}, Q^{\mathcal{C}} \leftarrow \mathsf{CLLZ.Enc}(\mathsf{pk}, y_b^{\mathcal{C}})$.
6. Ch samples keys K_1, K_2, K_3 and constructs the program P which hardcodes K_1, K_2, K_3. It then prepares $\rho = (\{|A_{is_i,s'_i}\rangle\}_{i\in\ell_0}, \mathsf{iO}(P))$ and sends to $\mathcal{A}$.
7. $\mathcal{A}$ on receiving $(\rho, y_b^{\mathcal{B}}, y_b^{\mathcal{C}})$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
8. Ch then generates $x^{\mathcal{B}}_{\mathsf{trigger}}, x^{\mathcal{C}}_{\mathsf{trigger}} \in \{0,1\}^n$ as follows:
   (a) Let $x^{\mathcal{B}}_{\mathsf{trigger}_1} = F_2(K_2, x_0^{\mathcal{B}} \| Q^{\mathcal{B}})$ and $x^{\mathcal{B}}_{\mathsf{trigger}_2} = F_3(K_3, x^{\mathcal{B}}_{\mathsf{trigger}_1})$. Let $x^{\mathcal{B}}_{\mathsf{trigger}} = x_0^{\mathcal{B}} \| x^{\mathcal{B}}_{\mathsf{trigger}_1} \| x^{\mathcal{B}}_{\mathsf{trigger}_2}$.
   (b) Let $x^{\mathcal{C}}_{\mathsf{trigger}_1} = F_2(K_2, x_0^{\mathcal{C}} \| Q^{\mathcal{C}})$ and $x^{\mathcal{C}}_{\mathsf{trigger}_2} = F_3(K_3, x^{\mathcal{C}}_{\mathsf{trigger}_1})$. Let $x^{\mathcal{C}}_{\mathsf{trigger}} = x_0^{\mathcal{C}} \| x^{\mathcal{C}}_{\mathsf{trigger}_1} \| x^{\mathcal{C}}_{\mathsf{trigger}_2}$.
9. Apply $(\mathcal{B}(x^{\mathcal{B}}_{\mathsf{trigger}}, \cdot) \otimes \mathcal{C}(x^{\mathcal{C}}_{\mathsf{trigger}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.
10. Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$, else 0.

Claim 24. *The output distributions of the hybrids* Hybrid_2 *and* Hybrid_3 *are identically distributed.*

Proof. The proof is identical to the proof of indistinguishability of Hybrid_2 and Hybrid_3 in the proof of [CLLZ21, Theorem 7.12]. □

Finally we give a reduction from Hybrid_3 to the indistinguishability from random anti-piracy experiment (fig. 34) for CLLZ *post-processing* single-decryptor encryption scheme, where CLLZ single decryptor encryption is the one given in fig. 4, for more details see [CLLZ21, Construction 1, Section 6.3, pg. 39]. Let $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ be an adversary in Hybrid_3 above. Consider the following non-local adversary $(\mathcal{R}_{\mathcal{A}}, \mathcal{R}_{\mathcal{B}}, \mathcal{R}_{\mathcal{C}})$:

1. $\mathcal{R}_{\mathcal{A}}$ samples $y_0^{\mathcal{B}}, y_1^{\mathcal{B}}, y_0^{\mathcal{C}}, y_1^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^m$.
2. $\mathcal{R}_{\mathcal{A}}$ gets the quantum decryptor $\{|A_{is_i,s'_i}\rangle\}_{i\in\ell_0}$ and a public key $\mathsf{pk} = (R_i^0, R_i^1)$ from Ch, the challenger in the correlated challenge SDE anti-piracy experiment (see fig. 35) for the CLLZ SDE scheme.
3. $\mathcal{R}_{\mathcal{A}}$ samples K_1, K_2, K_3 and prepares the circuit P using R_i^0, R_i^1 and the keys K_1, K_2, K_3. Let $\rho = \{|A_{is_i,s'_i}\rangle\}_{i\in\ell_0}, \mathsf{iO}(P))$.

4. $\mathcal{R}_{\mathcal{A}}$ samples a bit $d \xleftarrow{\$} \{0,1\}$and runs $\mathcal{A}$ on $(\rho, y_d^{\mathcal{B}}, y_d^{\mathcal{C}})$ and gets back the output $\sigma_{\mathcal{B},\mathcal{C}}$.

5. $\mathcal{R}_{\mathcal{A}}$ sends $(K_1, K_2, K_3, d, \sigma_{\mathcal{B}})$ to $\mathcal{R}_{\mathcal{B}}$ and $(K_1, K_2, K_3, d, \sigma_{\mathcal{C}})$ to $\mathcal{R}_{\mathcal{C}}$.

6. $\mathcal{R}_{\mathcal{B}}$ on receiving $(c^{\mathcal{B}}, (x_0^{\mathcal{B}}, T^{\mathcal{B}}))$ as the challenge cipher text from Ch as the challenge ciphertext and $K_1, K_2, K_3, d, \sigma_{\mathcal{B}}$ from $\mathcal{R}_{\mathcal{A}}$, does the following:

   (a) $\mathcal{R}_{\mathcal{B}}$ generates the circuit $Q^{\mathcal{B}}$ which on any input x_0 generates $r \leftarrow T^{\mathcal{B}}(x_0)$ and if the output is $\perp$ outputs $\perp$, else computes $\mathsf{DecPostProcess}(c^{\mathcal{B}}, r)$ and if the outcome is 0, output $y_0^{\mathcal{B}}$, else output $y_1^{\mathcal{B}}$. $\mathcal{R}_{\mathcal{B}}$ generates $\tilde{Q}^{\mathcal{B}} \leftarrow \mathsf{iO}(Q^{\mathcal{B}})$.

   (b) $\mathcal{R}_{\mathcal{B}}$ constructs $x_{\mathsf{trigger}}^{\mathcal{B}}$ as follows. Let $x_{\mathsf{trigger}_1}^{\mathcal{B}} = F_2(K_2, x_0^{\mathcal{B}} \| \widetilde{Q}^{\mathcal{B}})$ and $x_{\mathsf{trigger}_2}^{\mathcal{B}} = F_3(K_3, x_{\mathsf{trigger}_1}^{\mathcal{B}})$. Let $x_{\mathsf{trigger}}^{\mathcal{B}} = x_0^{\mathcal{B}} \| x_{\mathsf{trigger}_1}^{\mathcal{B}} \| x_{\mathsf{trigger}_2}^{\mathcal{B}}$.

   (c) $\mathcal{R}_{\mathcal{B}}$ runs $\mathcal{B}$ on $(x_{\mathsf{trigger}}^{\mathcal{B}}, \sigma_{\mathcal{B}})$ to get an output $b^{\mathcal{B}}$.

   (d) $\mathcal{R}_{\mathcal{B}}$ outputs $b^{\mathcal{B}} \oplus d$.

7. Similarly, $\mathcal{R}_{\mathcal{C}}$ on receiving $(c^{\mathcal{C}}, (x_0^{\mathcal{C}}, T^{\mathcal{C}}))$ as the challenge cipher text from Ch and $K_1, K_2, K_3, d, \sigma_{\mathcal{C}}$ from $\mathcal{R}_{\mathcal{A}}$, does the following:

   (a) $\mathcal{R}_{\mathcal{C}}$ generates the circuit $Q^{\mathcal{C}}$ which on any input x_0 generates $r \leftarrow T^{\mathcal{C}}(x_0)$ and if the output is $\perp$ outputs $\perp$, else computes $\mathsf{DecPostProcess}(c^{\mathcal{B}}, r)$ and if the outcome is 0, output $y_0^{\mathcal{C}}$, else output $y_1^{\mathcal{C}}$. $\mathcal{R}_{\mathcal{C}}$ generates $\tilde{Q}^{\mathcal{C}} \leftarrow \mathsf{iO}(Q^{\mathcal{C}})$.

   (b) $\mathcal{R}_{\mathcal{C}}$ constructs $x_{\mathsf{trigger}}^{\mathcal{C}}$ as follows. Let $x_{\mathsf{trigger}_1}^{\mathcal{C}} = F_2(K_2, x_0^{\mathcal{C}} \| \widetilde{Q}^{\mathcal{C}})$ and $x_{\mathsf{trigger}_2}^{\mathcal{C}} = F_3(K_3, x_{\mathsf{trigger}_1}^{\mathcal{C}})$. Let $x_{\mathsf{trigger}}^{\mathcal{C}} = x_0^{\mathcal{C}} \| x_{\mathsf{trigger}_1}^{\mathcal{C}} \| x_{\mathsf{trigger}_2}^{\mathcal{C}}$.

   (c) $\mathcal{R}_{\mathcal{C}}$ runs $\mathcal{C}$ on $(x_{\mathsf{trigger}}^{\mathcal{C}}, \sigma_{\mathcal{C}})$ to get an output $b^{\mathcal{C}}$.

   (d) $\mathcal{R}_{\mathcal{C}}$ outputs $b^{\mathcal{C}} \oplus d$.

Note that the functionality of $Q^{\mathcal{B}}$ and $Q^{\mathcal{C}}$ are the same as that of $W^{\mathcal{B}}, W^{\mathcal{C}}$ in the ciphertexts $(x_0^{\mathcal{B}}, W^{\mathcal{B}})$ and $(x_0^{\mathcal{C}}, W^{\mathcal{C}})$ obtained by running $\mathsf{CLLZ.Enc}(\mathsf{pk}, \cdot)$ algorithm on $y_b^{\mathcal{B}}$ and $y_b^{\mathcal{C}}$ with $x_0^{\mathcal{B}}$ and $x_0^{\mathcal{C}}$ as the randomness respectively. Note that in Hybrid_3, $\mathcal{B}$ (and similarly, $\mathcal{C}$) needs to distinguish between the following two inputs: a random string $y^{\mathcal{B}}$ along with either a triggered input $x^{\mathcal{B}}$ encoding $y^{\mathcal{B}}$ which is also the view of the inside adversary in the reduction above in the event $b = d$ in the simulated experiment; or a triggered input $x^{\mathcal{B}}$ encoding $\tilde{y}^{\mathcal{B}}$ random string where $\tilde{y}^{\mathcal{B}} \xleftarrow{\$}$ sampled independent of $y^{\mathcal{B}}$, which is the view of the inside adversary in the reduction above in the event $b \neq d$ in the simulated experiment. Therefore, by the iO guarantees, the view of the inside $\mathcal{A}, \mathcal{B}, \mathcal{C}$ is the same as that in Hybrid_3.

□

Proof of Proposition 21. The proof is the same as the proof for Proposition 20 up to minor changes.

We will start with a series of hybrids. The changes are marked in blue.

$\underline{\mathsf{Hybrid}_0}$: Same as $\mathsf{PreponedExpt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathsf{CP},\mathcal{D}_{\mathcal{X}}}(1^\lambda)$ (see Game 7) where $\mathcal{D} = \mathsf{Id}_{\mathcal{U}}$ (see the definition in Definition 19) for the CLLZ copy-protection scheme see Figure 8.

1. Ch samples $K_1 \leftarrow \mathsf{PRF.Gen}(1^\lambda)$ and generates $\rho = (\{|A_{i s_i, s'_i}\rangle\}_{i \in \ell_0}, \mathsf{iO}(P)) \leftarrow \mathsf{CLLZ.QKeyGen}(K_1)$, and sends ρ to $\mathcal{A}$. P has K_1, K_2, K_3 hardcoded in it where K_2, K_3 are the secondary keys.

2. Ch generates $x \xleftarrow{\$} \{0,1\}^n$, where $x = x_0\|x_1\|x_2$ and computes $y_0 \leftarrow \mathsf{PRF.Eval}(K_1, x)$.

3. Ch also samples $y_1 \xleftarrow{\$} \{0,1\}^m$.

4. Ch samples $b \xleftarrow{\$} \{0,1\}$, and sends $\mathcal{A}$, (ρ, y_b, y_b).

5. $\mathcal{A}$ on receiving (ρ, y_b, y_b) produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.

6. Apply $(\mathcal{B}(x,\cdot) \otimes \mathcal{C}(x,\cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.

7. Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$, else 0.

$\underline{\mathsf{Hybrid}_1}$: We modify the sampling procedure of the challenge input x.

1. Ch samples $K_1 \leftarrow \mathsf{PRF.Gen}(1^\lambda)$ and generates $\rho = (\{|A_{i s_i, s'_i}\rangle\}_{i\in\ell_0}, \mathsf{iO}(P)) \leftarrow \mathsf{CLLZ.QKeyGen}(K_1)$, and sends ρ to $\mathcal{A}$. P has K_1, K_2, K_3 hardcoded in it where K_2, K_3 are the secondary keys.

2. Ch generates $x \xleftarrow{\$} \{0,1\}^n$, where $x = x_0\|x_1\|x_2$ and computes $y_0 \leftarrow \mathsf{PRF.Eval}(K_1, x)$.

3. Ch also samples $y_1 \xleftarrow{\$} \{0,1\}^m$.

4. Ch also computes $x_{\mathsf{trigger}} \leftarrow \mathsf{Gen\text{-}Trigger}(x_0, y_0, K_2, K_3, \{A_{i s_i, s'_i}\}_{i\in\ell_0})$.

5. Ch also samples $y_1 \xleftarrow{\$} \{0,1\}^m$.

6. Ch samples $b \xleftarrow{\$} \{0,1\}$, and sends $\mathcal{A}$, (ρ, y_b, y_b).

7. $\mathcal{A}$ on receiving (ρ, y_b, y_b) produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.

8. Apply $(\mathcal{B}($ ~~x~~ x_{trigger} $,\cdot) \otimes \mathcal{C}($ ~~x~~ x_{trigger} $,\cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.

9. Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$, else 0.

Hybrid_1 is computationally indistinguishable from Hybrid_0 due to [CLLZ21, Lemma 7.17]. The same arguments via [CLLZ21, Lemma 7.17] were made in showing the indistinguishability between hybrids Hybrid_0 and Hybrid_1 in the proof of [CLLZ21, Theorem 7.12].

Claim 25. *Assuming the security of* PRF*, hybrids* Hybrid_0 *and* Hybrid_1 *are computationally indistinguishable.*

Proof. The proof is identical to the proof of indistinguishability of Hybrid_0 and Hybrid_1 in the proof of [CLLZ21, Theorem 7.12]. □

$\underline{\mathsf{Hybrid}_2}$: We modify the generation of the outputs y_0.

1. Ch samples $K_1 \leftarrow \mathsf{PRF.Gen}(1^\lambda)$ and generates $\rho = (\{|A_{i s_i, s'_i}\rangle\}_{i\in\ell_0}, \mathsf{iO}(P)) \leftarrow \mathsf{CLLZ.QKeyGen}(K_1)$, and sends ρ to $\mathcal{A}$. P has K_1, K_2, K_3 hardcoded in it where K_2, K_3 are the secondary keys.

2. Ch generates $x \xleftarrow{\$} \{0,1\}^n$, where $x = x_0\|x_1\|x_2$, and ~~computes $y_0 \leftarrow \mathsf{PRF.Eval}(K_1, x)$~~ samples $y_0 \xleftarrow{\$} \{0,1\}^m$.

3. Ch also computes $x_{\mathsf{trigger}} \leftarrow \mathsf{Gen\text{-}Trigger}(x_0, y_0, K_2, K_3, \{A_{i s_i, s'_i}\}_{i \in \ell_0})$.

4. Ch also samples $y_1 \xleftarrow{\$} \{0,1\}^m$.

5. Ch samples $b \xleftarrow{\$} \{0,1\}$ and sends $\mathcal{A}$ (ρ, y_b, y_b).

6. $\mathcal{A}$ on receiving (ρ, y_b, y_b) produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.

7. Apply $(\mathcal{B}(x_{\mathsf{trigger}}, \cdot) \otimes \mathcal{C}(x_{\mathsf{trigger}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.

8. Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$, else 0.

Hybrid_2 is statistically indistinguishable from Hybrid_1 due to the extractor properties of the primary PRF family. For more details, refer to the proof of see [CLLZ21, Theorem 7.12].

Claim 26. *Assuming the extractor properties of* PRF*, hybrids* Hybrid_1 *and* Hybrid_2 *are statistically indistinguishable.*

Proof. The proof is identical to the proof of indistinguishability of Hybrid_1 and Hybrid_2 in the proof of [CLLZ21, Theorem 7.12]. □

<u>Hybrid_3</u>: This hybrid is a reformulation of Hybrid_2.

1. Ch samples $\{A_{i s_i, s'_i}\}_{i \in \ell_0}$ and generates $\{|A_{i s_i, s'_i}\rangle\}_{i \in \ell_0}$, and treats it as the quantum decryption key for the CLLZ single-decryptor encryption scheme (see fig. 4), where the secret key is $\{A_{i s_i, s'_i}\}_{i \in \ell_0}$. Ch also generates $\mathsf{pk} = \{R_i^0, R_i^1\}_{i \in \ell_0}$, where for every $i \in [\ell_0]$, $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i^{\perp} + s'_i)$.

2. Ch ~~generates $x \xleftarrow{\$} \{0,1\}^n$, where $x = x_0 \| x_1 \| x_2$ and~~ samples $y_0 \xleftarrow{\$} \{0,1\}^m$.

3. ~~Ch also computes $x_{\mathsf{trigger}} \leftarrow \mathsf{Gen\text{-}Trigger}(x_0, y_0, K_2, K_3, \{A_{i s_i, s'_i}\}_{i \in \ell_0})$.~~

4. Ch also samples $y_1 \xleftarrow{\$} \{0,1\}^m$.

5. Ch samples $b \xleftarrow{\$} \{0,1\}$, and generates $x_0, Q \leftarrow \mathsf{CLLZ.Enc}(\mathsf{pk}, y_b)$.

6. Ch samples keys K_1, K_2, K_3 and constructs the program P which hardcodes K_1, K_2, K_3. It then prepares $\rho = (\{|A_{i s_i, s'_i}\rangle\}_{i \in \ell_0}, \mathsf{iO}(P))$ and sends to $\mathcal{A}$.

7. $\mathcal{A}$ on receiving (ρ, y_b, y_b) produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.

8. Ch then generates $x_{\mathsf{trigger}} \in \{0,1\}^n$ as follows: Let $x_{\mathsf{trigger}_1} = F_2(K_2, x_0 \| Q^{\mathcal{B}})$ and $x_{\mathsf{trigger}_2} = F_3(K_3, x_{\mathsf{trigger}_1})$. Let $x^{\mathcal{B}}_{\mathsf{trigger}} = x_0 \| x_{\mathsf{trigger}_1} \| x_{\mathsf{trigger}_2}$.

9. Apply $(\mathcal{B}(x^{\mathcal{B}}_{\mathsf{trigger}}, \cdot) \otimes \mathcal{C}(x^{\mathcal{C}}_{\mathsf{trigger}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.

10. Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$, else 0.

Claim 27. *The output distributions of the hybrids* Hybrid_2 *and* Hybrid_3 *are identically distributed.*

Proof. The proof is identical to the proof of indistinguishability of Hybrid_2 and Hybrid_3 in the proof of [CLLZ21, Theorem 7.12]. □

Finally we give a reduction from Hybrid_3 to the indistinguishability from random anti-piracy experiment (fig. 34) for CLLZ *post-processing* single-decryptor encryption scheme, where CLLZ single decryptor encryption is the one given in fig. 4, for more details see [CLLZ21, Construction 1, Section 6.3, pg. 39]. Let $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ be an adversary in Hybrid_3 above. Consider the following non-local adversary $(\mathcal{R}_\mathcal{A}, \mathcal{R}_\mathcal{B}, \mathcal{R}_\mathcal{C})$:

1. $\mathcal{R}_\mathcal{A}$ samples $y_0, y_1 \xleftarrow{\$} \{0,1\}^m$.
2. $\mathcal{R}_\mathcal{A}$ gets the quantum decryptor $\{|A_{i s_i, s'_i}\rangle\}_{i \in \ell_0}$ and a public key $\mathsf{pk} = (R_i^0, R_i^1)$ from Ch, the challenger in the correlated challenge SDE anti-piracy experiment (see fig. 35) for the CLLZ SDE scheme.
3. $\mathcal{R}_\mathcal{A}$ samples K_1, K_2, K_3 and prepares the circuit P using R_i^0, R_i^1 and the keys K_1, K_2, K_3. Let $\rho = \{|A_{i s_i, s'_i}\rangle\}_{i \in \ell_0}, \mathsf{iO}(P))$.
4. $\mathcal{R}_\mathcal{A}$ samples a bit $d \xleftarrow{\$} \{0,1\}$and runs $\mathcal{A}$ on (ρ, y_d, y_d) and gets back the output $\sigma_{\mathcal{B},\mathcal{C}}$.
5. $\mathcal{R}_\mathcal{A}$ samples a random string $s \xleftarrow{\$}$ of appropriate length as required by $\mathcal{B}$ and $\mathcal{C}$ to run the iO compiler.
6. $\mathcal{R}_\mathcal{A}$ sends $(K_1, K_2, K_3, d, s, \sigma_\mathcal{B})$ to $\mathcal{R}_\mathcal{B}$ and $(K_1, K_2, K_3, d, s, \sigma_\mathcal{C})$ to $\mathcal{R}_\mathcal{C}$.
7. $\mathcal{R}_\mathcal{B}$ on receiving $(c, (x_0, T))$ as the challenge cipher text from Ch as the challenge ciphertext and $K_1, K_2, K_3, d, s, \sigma_\mathcal{B}$ from $\mathcal{R}_\mathcal{A}$, does the following:
   (a) $\mathcal{R}_\mathcal{B}$ generates the circuit Q which on any input x_0 generates $r \leftarrow T(x_0)$ and if the output is $\perp$ outputs $\perp$, else computes $\mathsf{DecPostProcess}(c, r)$ and if the outcome is 0, output y_0, else output y_1. $\mathcal{R}_\mathcal{B}$ generates $\tilde{Q} \leftarrow \mathsf{iO}(Q; s)$.
   (b) $\mathcal{R}_\mathcal{B}$ constructs x_{trigger} as follows. Let $x_{\mathsf{trigger}_1} = F_2(K_2, x_0 \| \widetilde{Q})$ and $x_{\mathsf{trigger}_2} = F_3(K_3, x_{\mathsf{trigger}_1})$. Let $x_{\mathsf{trigger}} = x_0 \| x_{\mathsf{trigger}_1} \| x_{\mathsf{trigger}_2}$.
   (c) $\mathcal{R}_\mathcal{B}$ runs $\mathcal{B}$ on $(x_{\mathsf{trigger}}, \sigma_\mathcal{B})$ to get an output $b^\mathcal{B}$.
   (d) $\mathcal{R}_\mathcal{B}$ outputs $b^\mathcal{B} \oplus d$.
8. Similarly, $\mathcal{R}_\mathcal{C}$ on receiving $(c, (x_0, T))$ as the challenge cipher text from Ch and $K_1, K_2, K_3, d, s, \sigma_\mathcal{C}$ from $\mathcal{R}_\mathcal{A}$, does the following:
   (a) $\mathcal{R}_\mathcal{C}$ generates the circuit Q which on any input x_0 generates $r \leftarrow T(x_0)$ and if the output is $\perp$ outputs $\perp$, else computes $\mathsf{DecPostProcess}(c, r)$ and if the outcome is 0, output y_0, else output y_1. $\mathcal{R}_\mathcal{C}$ generates $\tilde{Q} \leftarrow \mathsf{iO}(Q; s)$.
   (b) $\mathcal{R}_\mathcal{C}$ constructs x_{trigger} as follows. Let $x_{\mathsf{trigger}_1} = F_2(K_2, x_0 \| \widetilde{Q})$ and $x_{\mathsf{trigger}_2} = F_3(K_3, x_{\mathsf{trigger}_1})$. Let $x_{\mathsf{trigger}} = x_0 \| x_{\mathsf{trigger}_1} \| x_{\mathsf{trigger}_2}$.
   (c) $\mathcal{R}_\mathcal{B}$ runs $\mathcal{B}$ on $(x_{\mathsf{trigger}}, \sigma_\mathcal{B})$ to get an output $b^\mathcal{C}$.
   (d) $\mathcal{R}_\mathcal{C}$ outputs $b^\mathcal{C} \oplus d$.

Note that the functionality of Q is the same as that of W in the cipher text (x_0, W) obtained by running $\mathsf{CLLZ.Enc}(\mathsf{pk}, \cdot)$ algorithm on y_b with x_0 as the randomness. Note that in Hybrid_3, $\mathcal{B}$ (and similarly, $\mathcal{C}$) needs to distinguish between the following two inputs: a random string y along with either a triggered input x encoding y which is also the view of the inside adversary in the reduction above in the event $b = d$ in the simulated experiment; or a triggered input x encoding $\tilde{y}$ random string where $\tilde{y} \xleftarrow{\$}$ sampled independent of y, which is the view of the inside adversary in the reduction above in the event $b \neq d$ in the simulated experiment. Therefore, by the iO guarantees, the view of the inside $\mathcal{A}, \mathcal{B}, \mathcal{C}$ is the same as that in Hybrid_3.

□

5.3 UPO for Keyed Circuits from Copy-Protection with Preponed Security

Theorem 28. *Assuming Conjecture 14, the existence of post-quantum sub-exponentially secure* iO *and one-way functions, and the quantum hardness of Learning-with-errors problem (LWE), there is a construction of unclonable puncturable obfuscation satisfying* $\mathcal{U}$*-generalized* UPO *security (see Definition 9), for any generalized keyed puncturable circuit class* $\mathfrak{C}$ *in* $\mathsf{P/poly}$*, see Section 3.1.1.*

Proof. The proof follows by combining Lemma 30 and theorem 31. □

Theorem 29. *Assuming Conjecture 13, the existence of post-quantum sub-exponentially secure* iO *and one-way functions, and the quantum hardness of Learning-with-errors problem (LWE), there is a construction of unclonable puncturable obfuscation satisfying* $\mathsf{Id}_{\mathcal{U}}$*-generalized* UPO *security (see Definition 9), for any generalized keyed puncturable circuit class* $\mathfrak{C}$ *in* $\mathsf{P/poly}$*, see Section 3.1.1.*

Proof. The proof follows by combining Lemma 30 and theorem 32. □

The construction is as follows. In the construction given in Figure 10, the PRF family $(\mathsf{KeyGen}, \mathsf{Eval})$ satisfies the requirements as in [CLLZ21] and has input length $n(\lambda)$ and output length m; PRG is a length-doubling injective pseudorandom generator with input length m.

Lemma 30. *The construction given in Figure 10 satisfies* $(1 - \mathsf{negl})$*-*UPO *correctness for any generalized puncturable keyed circuit class in* $\mathsf{P/poly}$ *for some negligible function* negl*.*

Proof. Let W be the circuit that is obfuscated, and let the resulting obfuscated state be $\rho = (\{\{|A_{i s_i, s'_i}\rangle\}_i\}, \tilde{C}, \mathsf{iO}(D))$. We will show that for every input $x = (x_0, x_1, x_2)$, the Eval algorithm on (ρ, x) outputs $W(x)$ except with negligible probability. Let $|\phi_x\rangle$ be the state obtained after running the Hadamard operation on $\{\{|A_{i s_i, s'_i}\rangle\}_i\}$ (see Item 2 of the Eval algorithm in Figure 10). It is easy to check that for every input x, by the correctness of CLLZ copy-protection, running $\tilde{C}$ that is generated as $\tilde{C} \leftarrow \mathsf{iO}(C)$ on $(x, |\phi_x\rangle$ in superposition, and then measuring the output register results in y which is equal to $\mathsf{PRG}(\mathsf{PRF.Eval}(k, x))$, except with negligible probability. By the almost as good as new lemma [Aar16], this would mean that the resulting quantum state σ which is negligibly close to $|\psi_x\rangle\langle\psi_x|$ in trace distance. Hence, running C on σ in Item 4 and inside $\mathsf{iO}(D)$ in superposition and then checking if the output is equal to y in superposition (see Item 4 of the $\mathsf{Eval}()$ algorithm in Figure 10), must succeed and $\mathsf{iO}(D)$ will output $W(x)$, except with negligible probability. Therefore, except with negligible probability, $\mathsf{Eval}(\rho, x)$ outputs $W(x)$. □

Theorem 31. *Assuming Conjecture 14, post-quantum sub-exponentially secure* iO *and one-way functions, and the quantum hardness of Learning-with-errors problem (LWE), the construction given in Figure 10 satisfies* $\mathcal{U}$*-generalized unclonable puncturable obfuscation security (see Section 3.1.1) for any generalized puncturable keyed circuit class in* $\mathsf{P/poly}$*.*

Assumes: PRF family (KeyGen, Eval) with same properties as needed in [CLLZ21], PRG, CLLZ copy-protection scheme (CopyProtect, Eval).

$\mathsf{Obf}(1^\lambda, W)$:

1. Sample a random key $k \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$.
2. Compute $\mathsf{iO}(P), \{\{|A_{is_i,s'_i}\rangle\}_i\} \leftarrow \mathsf{CLLZ.CopyProtect}(k)$.
3. Compute $\tilde{C} \leftarrow \mathsf{iO}(C)$ where $C = \mathsf{PRG} \cdot \mathsf{iO}(P)$.
4. Compute $\mathsf{iO}(D)$ where D takes as input x, v, y, and runs C on x, v to get y' and outputs $\perp$ if $y' \neq y$ or $y' = \perp$, else it runs the circuit W on x to output $W(x)$.
5. Output $\rho = (\{\{|A_{is_i,s'_i}\rangle\}_i\}, \tilde{C}, \mathsf{iO}(D))$.

$\mathsf{Eval}(\rho, x)$

1. Interprete $\rho = (\{\{|A_{is_i,s'_i}\rangle\}_i\}, \tilde{C}, \mathsf{iO}(D))$.
2. Let $x = x_0 \| x_1 \| x_2$, where $x_0 = \ell_0$. For every $i \in [\ell_0]$, if $x_{0,i} = 1$ apply $H^{\otimes n}$ on $|A_{is_i,s'_i}\rangle$. Let the resulting state be $|\psi_x\rangle$.
3. Run the circuit $\tilde{C}$ in superposition on the input registers (X, V) with the initial state $(x, |\psi_x\rangle)$ and then measure the output register to get an output y. Let the resulting state quantum state on register V be σ.
4. Run $\mathsf{iO}(D)$ on the registers X, V, Y in superposition where registers X, Y are initialized to classical values x, y and then measure the output register to get an output z. Output z.

Figure 10: Construction of a UPO scheme.

Proof. The proof follows by combining Lemma 33, Proposition 20, and theorem 15, and the observation that the quantum hardness of LWE implies post-quantum one-way functions. □

Theorem 32. *Assuming Conjecture 13, the existence of post-quantum sub-exponentially secure* iO *and one-way functions, and the quantum hardness of Learning-with-errors problem (LWE), the construction given in Figure 10 satisfies* $\mathsf{Id}_{\mathcal{U}}$*-generalized unclonable puncturable obfuscation security (see Section 3.1.1) for any generalized puncturable keyed circuit class in* P/poly.

Proof. The proof follows by combining Lemma 34, Proposition 21, and theorem 16, and the observation that the quantum hardness of LWE implies post-quantum one-way functions. □

Lemma 33. *Assuming the existence of post-quantum* iO*, one-way functions, and that* CLLZ *copy protection construction for* PRF*s given in Figure 8, satisfies* $\mathcal{U}$*-preponed security (defined in Definition 19, the construction given in Figure 10 for* $\mathcal{W}$ *satisfies* $\mathcal{U}$*-generalized* UPO *security guarantee (see Section 3.1.1), for any puncturable keyed circuit class* $\mathcal{W} = \{\{W_s\}_{s \in \mathcal{K}_\lambda}\}_\lambda$ *in* P/poly.

Lemma 34. *Assuming the existence of post-quantum* iO*, one-way functions, and that* CLLZ *copy protection construction for* PRF*s given in Figure 8, satisfies* $\mathsf{Id}_{\mathcal{U}}$*-preponed security (defined in Definition 19), the construction given in Figure 10 for* $\mathcal{W}$ *satisfies* $\mathsf{Id}_{\mathcal{U}}$*-generalized* UPO *security guarantee (see Section 3.1.1), for any puncturable keyed circuit class* $\mathcal{W} = \{\{W_s\}_{s \in \mathcal{K}_\lambda}\}_\lambda$ *in* P/poly.

Proof of Lemma 33. We mark the changes in blue.

Hybrid$_0$:
Same as the security experiment given in fig. 3 with $\mathcal{D}_\mathcal{X} = \mathcal{U}$ as mentioned in the lemma.

1. $\mathcal{A}$ sends a key $s \in \mathcal{K}_\lambda$ and functions $\mu_\mathcal{B}$ and $\mu_\mathcal{C}$ to Ch.
2. Ch samples $x^\mathcal{B}, x^\mathcal{C} \xleftarrow{\$} \{0,1\}^n$.
3. Ch samples $k \leftarrow \mathsf{KeyGen}$, and generates $\mathsf{iO}(P), \{|A_{is_i,s'_i}\rangle\}_i \leftarrow \mathsf{CLLZ.CopyProtect}(1^\lambda, k)$.
4. Ch constructs $\tilde{C} \leftarrow \mathsf{iO}(C)$ where $C = \mathsf{PRG} \cdot \mathsf{iO}(P)$.
5. Ch constructs the circuit $\mathsf{iO}(D_0), \mathsf{iO}(D_1)$ where D_0, D_1 are as depicted in figs. 12 and 13.
6. Ch samples $b \xleftarrow{\$} \{0,1\}$ and sends $(\mathsf{iO}(C), \{|A_{is_i,s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ to $\mathcal{A}$.
7. $\mathcal{A}(\tilde{C}, \{|A_{is_i,s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ outputs a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
8. Apply $(\mathcal{B}(x^\mathcal{B}, \cdot) \otimes \mathcal{C}(x^\mathcal{C}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_\mathbf{B}, b_\mathbf{C})$.
9. Output 1 if $b_\mathbf{B} = b_\mathbf{C} = b$.

$\underline{P}$:

Hardcoded keys $K_1, K_2, K_3, R_i^0, R_i^1$ for every $i \in [\ell_0]$ On input $x = x_0\|x_1\|x_2$ and vectors $v = v_1, \ldots v_{\ell_0}$.

1. If $F_3(K_3, x_1) \oplus x_2 = x_0\|Q$ and $x_1 = F_2(K_2, x_0\|Q)$:
   Hidden trigger mode: Treat Q as a classical circuit and output $Q(v)$.
2. Otherwise, check if the following holds: for all $i \in \ell_0$, $R^{x_{0,i}}(v_i) = 1$ (where $x_{0,i}$ is the i^{th} coordinate of x_0).
   Normal mode: If so, output $F_1(K_1, x)$ where $F_1() = \mathsf{PRF.Eval}()$ is the primary pseudo-random function family that is being copy-protected. Otherwise output $\perp$.

Figure 11: Circuit P in Hybrid$_0$.

$\underline{\text{Hybrid}_1}$:

1. $\mathcal{A}$ sends a key $s \in \mathcal{K}_\lambda$ and functions $\mu_\mathcal{B}$ and $\mu_\mathcal{C}$ to Ch.
2. Ch samples $x^\mathcal{B}, x^\mathcal{C} \xleftarrow{\$} \{0,1\}^n$.
3. Ch samples $k \leftarrow \mathsf{KeyGen}$, and generates $\mathsf{iO}(P), \{|A_{is_i,s'_i}\rangle\}_i \leftarrow \mathsf{CLLZ.CopyProtect}(1^\lambda, k)$.
4. Ch constructs $\tilde{C} \leftarrow \mathsf{iO}(C)$ where $C = \mathsf{PRG} \cdot \mathsf{iO}(P)$.

$\underline{D_0}$:

Hardcoded keys W_s, C. On input: x, v, y.

1. Run $y' \leftarrow C(x, v)$.
2. If $y' \neq y$ or $y' = \perp$ output $\perp$.
3. If $y = y' \neq \perp$, output $W_s(x)$.

Figure 12: Circuit D_0 in Hybrid_0

$\underline{D_1}$:

Hardcoded keys $W_{s,x^{\mathcal{B}},x^{\mathcal{C}},\mu_{\mathcal{B}},\mu_{\mathcal{C}}}$,$C$. On input: x, v, y.

1. Run $y' \leftarrow C(x, v)$.
2. If $y' \neq y$ or $y' = \perp$ output $\perp$.
3. If $y = y' \neq \perp$, output $W_{s,x^{\mathcal{B}},x^{\mathcal{C}},\mu_{\mathcal{B}},\mu_{\mathcal{C}}}(x)$.

Figure 13: Circuit D_1 in Hybrid_0

5. Ch samples $y_0^{\mathcal{B}}, y_1^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^{2m}$.
6. Ch constructs the circuit $\mathsf{iO}(D_0), \mathsf{iO}(D_1)$ where D_0 and D_1 are as depicted in fig. 14 and fig. 13, respectively.
7. Ch samples $b \xleftarrow{\$} \{0,1\}$ and sends $(\mathsf{iO}(C), \{|A_{i s_i, s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ to $\mathcal{A}$.
8. $\mathcal{A}(\tilde{C}, \{|A_{i s_i, s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ outputs a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
9. Apply $(\mathcal{B}(x^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(x^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_{\mathbf{B}}, b_{\mathbf{C}})$.
10. Output 1 if $b_{\mathbf{B}} = b_{\mathbf{C}} = b$.

$\underline{\mathsf{Hybrid}_2}$:

1. $\mathcal{A}$ sends a key $s \in \mathcal{K}_\lambda$ and functions $\mu_{\mathcal{B}}$ and $\mu_{\mathcal{C}}$ to Ch.
2. Ch samples $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$.
3. Ch samples $k \leftarrow \mathsf{KeyGen}$, and generates $\mathsf{iO}(P), \{|A_{i s_i, s'_i}\rangle\}_i \leftarrow \mathsf{CLLZ.CopyProtect}(1^\lambda, k)$.

$\underline{D_0}$:

Hardcoded keys W_s,$\mu_{\mathcal{B}}, \mu_{\mathcal{C}}$,$C$. On input: x, v, y.

1. Run $y' \leftarrow C(x, v)$.
2. If $y' \neq y$ or $y' = \perp$ output $\perp$.
3. If $y = y' \neq \perp$ and $y \in \{y_0^{\mathcal{B}}, y_0^{\mathcal{C}}\}$:, output $g(x)$.
   (a) If $y = y_0^{\mathcal{B}}$ output $\mu_{\mathcal{B}}(x^{\mathcal{B}})$.
   (b) If $y = y_0^{\mathcal{C}}$ output $\mu_{\mathcal{C}}(x^{\mathcal{C}})$.
4. If $y = y' \neq \perp$ and $y \notin \{y_0^{\mathcal{B}}, y_0^{\mathcal{C}}\}$, output $W_s(x)$.

Figure 14: Circuit D_0 in Hybrid_1

4. Ch constructs $\tilde{C} \leftarrow \mathsf{iO}(C)$ where $C = \mathsf{PRG} \cdot \mathsf{iO}(P)$.
5. Ch samples ~~$y_0^{\mathcal{B}}, y_1^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^{2m}$~~ $y_1, y_2 \xleftarrow{\$} \{0,1\}^m$, and computes $y_0^{\mathcal{B}} \leftarrow \mathsf{PRG}(y_1), y_0^{\mathcal{C}} \leftarrow \mathsf{PRG}(y_2)$.
6. Ch constructs the circuit $\mathsf{iO}(D_0), \mathsf{iO}(D_1)$ where D_0 and D_1 are as depicted in figs. 13 and 14, respectively.
7. Ch samples $b \xleftarrow{\$} \{0,1\}$ and sends $(\mathsf{iO}(C), \{|A_{is_i,s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ to $\mathcal{A}$.
8. $\mathcal{A}(\tilde{C}, \{|A_{is_i,s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ outputs a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
9. Apply $(\mathcal{B}(x^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(x^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_{\mathbf{B}}, b_{\mathbf{C}})$.
10. Output 1 if $b_{\mathbf{B}} = b_{\mathbf{C}} = b$.

$\underline{\mathsf{Hybrid}_3}$:

1. $\mathcal{A}$ sends a key $s \in \mathcal{K}_\lambda$ and functions $\mu_{\mathcal{B}}$ and $\mu_{\mathcal{C}}$ to Ch.
2. Ch samples $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$.
3. Ch samples $k \leftarrow \mathsf{KeyGen}$, and generates $\mathsf{iO}(P), \{|A_{is_i,s'_i}\rangle\}_i \leftarrow \mathsf{CLLZ.CopyProtect}(1^\lambda, k)$.
4. Ch constructs $\tilde{C} \leftarrow \mathsf{iO}(C)$ where $C = \mathsf{PRG} \cdot \mathsf{iO}(P)$.
5. Ch samples $y_1, y_2 \xleftarrow{\$} \{0,1\}^m$, and computes $y_0^{\mathcal{B}} \leftarrow \mathsf{PRG}(y_1), y_0^{\mathcal{C}} \leftarrow \mathsf{PRG}(y_2)$.
6. Ch constructs the circuit $\mathsf{iO}(D_0), \mathsf{iO}(D_1)$ where D_0 and D_1 are as depicted in fig. 14 and fig. 15, respectively.

7. Ch samples $b \xleftarrow{\$} \{0,1\}$ and sends $(\mathsf{iO}(C), \{|A_{is_i,s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ to $\mathcal{A}$.
8. $\mathcal{A}(\tilde{C}, \{|A_{is_i,s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ outputs a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
9. Apply $(\mathcal{B}(x^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(x^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_{\mathbf{B}}, b_{\mathbf{C}})$.
10. Output 1 if $b_{\mathbf{B}} = b_{\mathbf{C}} = b$.

$\underline{D_1}$:

Hardcoded keys f, g, C. On input: x, v, y.

1. Run $y' \leftarrow C(x, v)$
2. If $y' \neq y$ or $y' = \bot$ output $\bot$.
3. ~~If $y = y' \neq \bot$, output $W_{s,x^{\mathcal{B}},x^{\mathcal{C}},\mu_{\mathcal{B}},\mu_{\mathcal{C}}}(x)$.~~
4. If $y = y' \neq \bot$ and $x \in \{x^{\mathcal{B}}, x^{\mathcal{C}}\}$:
   (a) If $x = x^{\mathcal{B}}$ output $\mu_{\mathcal{B}}(x^{\mathcal{B}})$.
   (b) If $x = x^{\mathcal{C}}$ output $\mu_{\mathcal{C}}(x^{\mathcal{C}})$.
5. If $y = y' \neq \bot$ and $x \notin \{x^{\mathcal{B}}, x^{\mathcal{C}}\}$, output $W_s(x)$.

Figure 15: Circuit D_1 in Hybrid_3

$\underline{\mathsf{Hybrid}_4}$:

1. $\mathcal{A}$ sends a key $s \in \mathcal{K}_\lambda$ and functions $\mu_{\mathcal{B}}$ and $\mu_{\mathcal{C}}$ to Ch.
2. Ch samples $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$.
3. Ch samples $k \leftarrow \mathsf{KeyGen}$, and runs the $\mathsf{CLLZ.CopyProtect}(1^\lambda, k)$ algorithm as follows:~~generates $\mathsf{iO}(P), \{|A_{is_i,s'_i}\rangle\}_i \leftarrow \mathsf{CLLZ.CopyProtect}(1^\lambda, k)$.~~[7]
   (a) Samples ℓ_0 coset states $|A_{is_i,s'_i}\rangle_i$ and construct $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i + s'_i)$ for every $i \in [\ell_0]$.
   (b) Samples keys K_2, K_3 from the respective secondary PRFs and use $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i + s'_i)$ along with k to construct P, as given in fig. 11.
4. Ch computes $y_1^{\mathcal{B}} = \mathsf{PRG}(\mathsf{PRF.Eval}(k, x^{\mathcal{B}})), y_1^{\mathcal{C}} = \mathsf{PRG}(\mathsf{PRF.Eval}(k, x^{\mathcal{C}}))$, and uses $y_1^{\mathcal{B}}, y_1^{\mathcal{C}}$ along with $R_i^0, R_i^1, \mathsf{iO}(P), \mathsf{PRG}$ to construct C as depicted in fig. 16.

[7]There is no change in this line compared to Hybrid_3, we only spell out the $\mathsf{CLLZ.CopyProtect}(1^\lambda, k)$ explicitly in order to use intermediate information in the next few steps.

5. Ch samples $y_1, y_2 \xleftarrow{\$} \{0,1\}^m$, and computes $y_0^{\mathcal{B}} \leftarrow \mathsf{PRG}(y_1), y_0^{\mathcal{C}} \leftarrow \mathsf{PRG}(y_2)$.
6. Ch constructs the circuit $\mathsf{iO}(D_0), \mathsf{iO}(D_1)$ where D_0 and D_1 are as depicted in figs. 14 and 15, respectively.
7. Ch samples $b \xleftarrow{\$} \{0,1\}$ and sends $(\mathsf{iO}(C), \{|A_{i s_i, s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ to $\mathcal{A}$.
8. $\mathcal{A}(\tilde{C}, \{|A_{i s_i, s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ outputs a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
9. Apply $(\mathcal{B}(x^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(x^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_{\mathbf{B}}, b_{\mathbf{C}})$.
10. Output 1 if $b_{\mathbf{B}} = b_{\mathbf{C}} = b$.

<u>C</u>:

Hardcoded keys $\mathsf{iO}(P), \mathsf{PRG}, x^{\mathcal{B}}, x^{\mathcal{C}}, y_1^{\mathcal{B}}, y_1^{\mathcal{C}}, k_{x^{\mathcal{B}}, x^{\mathcal{C}}1}, R_i^0, R_i^1$ for all $i \in \ell_0$ (where ℓ_0 is the number of coset states.). On input: x, v.

1. If $x \in (x^{\mathcal{B}}, x^{\mathcal{C}})$:
   (a) Check if $R_i^{x_{0,i}}(v_i) = 1$ for all $i \in \ell_0$, and reject otherwise.
   (b) If $x = x^{\mathcal{B}}$, output $y_1^{\mathcal{B}}$.
   (c) If $x = x^{\mathcal{C}}$, output $y_1^{\mathcal{C}}$.
2. If $x \notin \{x^{\mathcal{B}}, x^{\mathcal{C}}\}$, output $\mathsf{PRG}(\mathsf{iO}(P)(x))$.

Figure 16: Circuit C in Hybrid_4

<u>Hybrid_5</u>:

1. $\mathcal{A}$ sends a key $s \in \mathcal{K}_\lambda$ and functions $\mu_{\mathcal{B}}$ and $\mu_{\mathcal{C}}$ to Ch.
2. Ch samples $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$.
3. Ch samples $k \leftarrow \mathsf{KeyGen}$, and does the following:
   (a) Computes $k_{x^{\mathcal{B}}, x^{\mathcal{C}}} \leftarrow \mathsf{PRF.Puncture}(k, \{x^{\mathcal{B}}, x^{\mathcal{C}}\})$.
   (b) Samples ℓ_0 coset states $|A_{i s_i, s'_i}\rangle_i$ and construct $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i + s'_i)$ for every $i \in [\ell_0]$.
   (c) Samples keys K_2, K_3 from the respective secondary PRFs and use $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i + s'_i)$ along with $k_{x^{\mathcal{B}}, x^{\mathcal{C}}}$ to construct P, as given in fig. 11.
4. Ch computes $y_1^{\mathcal{B}} = \mathsf{PRG}(\mathsf{PRF.Eval}(k, x^{\mathcal{B}}))$, $y_1^{\mathcal{C}} = \mathsf{PRG}(\mathsf{PRF.Eval}(k, x^{\mathcal{C}}))$ and uses $y_1^{\mathcal{B}}, y_1^{\mathcal{C}}$ along with $R_i^0, R_i^1, \mathsf{iO}(P), \mathsf{PRG}$ to construct C as depicted in fig. 16.

5. Ch samples $y_1, y_2 \xleftarrow{\$} \{0,1\}^m$, and computes $y_0^{\mathcal{B}} \leftarrow \mathsf{PRG}(y_1), y_0^{\mathcal{C}} \leftarrow \mathsf{PRG}(y_2)$.
6. Ch constructs the circuit $\mathsf{iO}(D_0), \mathsf{iO}(D_1)$ where D_0 and D_1 are as depicted in figs. 14 and 15, respectively.
7. Ch samples $b \xleftarrow{\$} \{0,1\}$ and sends $(\mathsf{iO}(C), \{|A_{i s_i, s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ to $\mathcal{A}$.
8. $\mathcal{A}(\tilde{C}, \{|A_{i s_i, s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ outputs a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
9. Apply $(\mathcal{B}(x^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(x^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_{\mathbf{B}}, b_{\mathbf{C}})$.
10. Output 1 if $b_{\mathbf{B}} = b_{\mathbf{C}} = b$.

$\underline{\mathsf{Hybrid}_6}$:

1. $\mathcal{A}$ sends a key $s \in \mathcal{K}_\lambda$ and functions $\mu_{\mathcal{B}}$ and $\mu_{\mathcal{C}}$ to Ch.
2. Ch samples $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$.
3. Ch samples $k \leftarrow \mathsf{KeyGen}$, and does the following:
   (a) Computes $k_{x^{\mathcal{B}}, x^{\mathcal{C}}} \leftarrow \mathsf{PRF.Puncture}(k, \{x^{\mathcal{B}}, x^{\mathcal{C}}\})$.
   (b) Samples ℓ_0 coset states $|A_{i s_i, s'_i}\rangle_i$ and construct $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i + s'_i)$ for every $i \in [\ell_0]$.
   (c) Samples keys K_2, K_3 from the respective secondary PRFs and use $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i + s'_i)$ along with $k_{x^{\mathcal{B}}, x^{\mathcal{C}}}$ to construct P, as given in fig. 11.
4. Ch samples $u^{\mathcal{B}}, u^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^m$ and computes $y_1^{\mathcal{B}} = \mathsf{PRG}(u^{\mathcal{B}}), y_1^{\mathcal{C}} = \mathsf{PRG}(u^{\mathcal{C}})$ ~~Ch computes $y_1^{\mathcal{B}} = \mathsf{PRF.Eval}(k, x^{\mathcal{B}}), y_1^{\mathcal{C}} = \mathsf{PRF.Eval}(k, x^{\mathcal{C}})$~~ and uses $y_1^{\mathcal{B}}, y_1^{\mathcal{C}}$ along with $R_i^0, R_i^1, \mathsf{iO}(P), \mathsf{PRG}$ to construct C as depicted in fig. 16.
5. Ch samples $y_1, y_2 \xleftarrow{\$} \{0,1\}^m$, and computes $y_0^{\mathcal{B}} \leftarrow \mathsf{PRG}(y_1), y_0^{\mathcal{C}} \leftarrow \mathsf{PRG}(y_2)$.
6. Ch constructs the circuit $\mathsf{iO}(D_0), \mathsf{iO}(D_1)$ where D_0 and D_1 are as depicted in figs. 14 and 15, respectively.
7. Ch samples $b \xleftarrow{\$} \{0,1\}$ and sends $(\mathsf{iO}(C), \{|A_{i s_i, s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ to $\mathcal{A}$.
8. $\mathcal{A}(\tilde{C}, \{|A_{i s_i, s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ outputs a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
9. Apply $(\mathcal{B}(x^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(x^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_{\mathbf{B}}, b_{\mathbf{C}})$.
10. Output 1 if $b_{\mathbf{B}} = b_{\mathbf{C}} = b$.

$\underline{\mathsf{Hybrid}_7}$:

1. $\mathcal{A}$ sends a key $s \in \mathcal{K}_\lambda$ and functions $\mu_{\mathcal{B}}$ and $\mu_{\mathcal{C}}$ to Ch.
2. Ch samples $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$.
3. Ch samples $k \leftarrow \mathsf{KeyGen}$, and does the following:

(a) Computes $k_{x^{\mathcal{B}},x^{\mathcal{C}}} \leftarrow \mathsf{PRF.Puncture}(k, \{x^{\mathcal{B}}, x^{\mathcal{C}}\})$.

(b) Samples ℓ_0 coset states $|A_{i s_i, s'_i}\rangle_i$ and construct $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i + s'_i)$ for every $i \in [\ell_0]$.

(c) Samples keys K_2, K_3 from the respective secondary PRFs and use $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i + s'_i)$ along with $k_{x^{\mathcal{B}},x^{\mathcal{C}}}$ to construct P, as given in fig. 11.

4. Ch samples $y_1^{\mathcal{B}}, y_1^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^{2m}$ ~~Ch samples $u^{\mathcal{B}}, u^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^m$ and computes $y_1^{\mathcal{B}} = \mathsf{PRG}(u^{\mathcal{B}}), y_1^{\mathcal{C}} = \mathsf{PRG}(u^{\mathcal{C}})$~~ and uses $y_1^{\mathcal{B}}, y_1^{\mathcal{C}}$ along with $R_i^0, R_i^1, \mathsf{iO}(P), \mathsf{PRG}$ to construct C as depicted in fig. 16.

5. Ch samples $y_1, y_2 \xleftarrow{\$} \{0,1\}^m$, and computes $y_0^{\mathcal{B}} \leftarrow \mathsf{PRG}(y_1), y_0^{\mathcal{C}} \leftarrow \mathsf{PRG}(y_2)$.

6. Ch constructs the circuit $\mathsf{iO}(D_0), \mathsf{iO}(D_1)$ where D_0 and D_1 are as depicted in figs. 14 and 15, respectively.

7. Ch samples $b \xleftarrow{\$} \{0,1\}$ and sends $(\mathsf{iO}(C), \{|A_{i s_i, s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ to $\mathcal{A}$.

8. $\mathcal{A}(\tilde{C}, \{|A_{i s_i, s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ outputs a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.

9. Apply $(\mathcal{B}(x^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(x^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_{\mathbf{B}}, b_{\mathbf{C}})$.

10. Output 1 if $b_{\mathbf{B}} = b_{\mathbf{C}} = b$.

<u>Hybrid$_8$</u>:

1. $\mathcal{A}$ sends a key $s \in \mathcal{K}_\lambda$ and functions $\mu_{\mathcal{B}}$ and $\mu_{\mathcal{C}}$ to Ch.

2. Ch samples $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$.

3. Ch samples $k \leftarrow \mathsf{KeyGen}$, and does the following:

   (a) Computes $k_{x^{\mathcal{B}},x^{\mathcal{C}}} \leftarrow \mathsf{PRF.Puncture}(k, \{x^{\mathcal{B}}, x^{\mathcal{C}}\})$.

   (b) Samples ℓ_0 coset states $|A_{i s_i, s'_i}\rangle_i$ and construct $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i + s'_i)$ for every $i \in [\ell_0]$.

   (c) Samples keys K_2, K_3 from the respective secondary PRFs and use $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i + s'_i)$ along with $k_{x^{\mathcal{B}},x^{\mathcal{C}}}$ to construct P.

4. Ch samples $y_1^{\mathcal{B}}, y_1^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^{2m}$ and uses $y_1^{\mathcal{B}}, y_1^{\mathcal{C}}$ along with $R_i^0, R_i^1, \mathsf{iO}(P), \mathsf{PRG}$ to construct C as depicted in fig. 16.

5. Ch samples $y_1, y_2 \xleftarrow{\$} \{0,1\}^m$, and computes $y_0^{\mathcal{B}} \leftarrow \mathsf{PRG}(y_1), y_0^{\mathcal{C}} \leftarrow \mathsf{PRG}(y_2)$.

6. Ch constructs the circuit $\mathsf{iO}(D_0), \mathsf{iO}(D_1)$ where D_0 and D_1 are as depicted in fig. 14 and fig. 17, respectively.

7. Ch samples $b \xleftarrow{\$} \{0,1\}$ and sends $(\mathsf{iO}(C), \{|A_{i s_i, s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ to $\mathcal{A}$.

8. $\mathcal{A}(\tilde{C}, \{|A_{i s_i, s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ outputs a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.

9. Apply $(\mathcal{B}(x^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(x^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_{\mathbf{B}}, b_{\mathbf{C}})$.

$\underline{D_1}$:

Hardcoded keys f, g, C. On input: x, v, y.

1. Run $y' \leftarrow C(x, v)$
2. If $y' \neq y$ or $y' = \perp$ output $\perp$.
3. If $y = y' \neq \perp$ and $y \in \{y_1^{\mathcal{B}}, y_1^{\mathcal{C}}\}$~~$x \in \{x^{\mathcal{B}}, x^{\mathcal{C}}\}$~~:
   (a) If $y = y_1^{\mathcal{B}}$ ~~$x = x^{\mathcal{B}}$~~ output $\mu_{\mathcal{B}}(x^{\mathcal{B}})$.
   (b) If $y = y_1^{\mathcal{C}}$ ~~$x = x^{\mathcal{C}}$~~ output $\mu_{\mathcal{C}}(x^{\mathcal{C}})$.
4. If $y = y' \neq \perp$ and $y \notin \{y_1^{\mathcal{B}}, y_1^{\mathcal{C}}\}$~~$x \notin \{x^{\mathcal{B}}, x^{\mathcal{C}}\}$~~, output $W_s(x)$.

Figure 17: Circuit D_1 in Hybrid_8

10. Output 1 if $b_{\mathbf{B}} = b_{\mathbf{C}} = b$.

$\underline{\mathsf{Hybrid}_9}$:

1. $\mathcal{A}$ sends a key $s \in \mathcal{K}_\lambda$ and functions $\mu_{\mathcal{B}}$ and $\mu_{\mathcal{C}}$ to Ch.
2. Ch samples $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$.
3. Ch samples $k \leftarrow \mathsf{KeyGen}$, and does the following:
   (a) Computes $k_{x^{\mathcal{B}}, x^{\mathcal{C}}} \leftarrow \mathsf{PRF.Puncture}(k, \{x^{\mathcal{B}}, x^{\mathcal{C}}\})$.
   (b) Samples ℓ_0 coset states $|A_{i s_i, s'_i}\rangle_i$ and construct $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i + s'_i)$ for every $i \in [\ell_0]$.
   (c) Samples keys K_2, K_3 from the respective secondary PRFs and use $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i + s'_i)$ along with $k_{x^{\mathcal{B}}, x^{\mathcal{C}}}$ to construct P, as given in fig. 11.
4. Ch samples $u^{\mathcal{B}}, u^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^m$ and computes $y_1^{\mathcal{B}} = \mathsf{PRG}(u^{\mathcal{B}}), y_1^{\mathcal{C}} = \mathsf{PRG}(u^{\mathcal{C}})$ ~~Ch samples $y_1^{\mathcal{B}}, y_1^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^{2m}$~~ and uses $y_1^{\mathcal{B}}, y_1^{\mathcal{C}}$ along with $R_i^0, R_i^1, \mathsf{iO}(P), \mathsf{PRG}$ to construct C as depicted in fig. 16.
5. Ch samples $y_1, y_2 \xleftarrow{\$} \{0,1\}^m$, and computes $y_0^{\mathcal{B}} \leftarrow \mathsf{PRG}(y_1), y_0^{\mathcal{C}} \leftarrow \mathsf{PRG}(y_2)$.
6. Ch constructs the circuit $\mathsf{iO}(D_0), \mathsf{iO}(D_1)$ where D_0 and D_1 are as depicted in figs. 14 and 17, respectively.
7. Ch samples $b \xleftarrow{\$} \{0,1\}$ and sends $(\mathsf{iO}(C), \{|A_{i s_i, s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ to $\mathcal{A}$.
8. $\mathcal{A}(\tilde{C}, \{|A_{i s_i, s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ outputs a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.

9. Apply $(\mathcal{B}(x^{\mathcal{B}},\cdot)\otimes\mathcal{C}(x^{\mathcal{C}},\cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_{\mathbf{B}},b_{\mathbf{C}})$.
10. Output 1 if $b_{\mathbf{B}}=b_{\mathbf{C}}=b$.

Hybrid$_{10}$:

1. $\mathcal{A}$ sends a key $s\in\mathcal{K}_\lambda$ and functions $\mu_{\mathcal{B}}$ and $\mu_{\mathcal{C}}$ to Ch.
2. Ch samples $x^{\mathcal{B}},x^{\mathcal{C}}\xleftarrow{\$}\{0,1\}^n$.
3. Ch samples $k\leftarrow\mathsf{KeyGen}$, and does the following:
   (a) Computes $k_{x^{\mathcal{B}},x^{\mathcal{C}}}\leftarrow\mathsf{PRF.Puncture}(k,\{x^{\mathcal{B}},x^{\mathcal{C}}\})$.
   (b) Samples ℓ_0 coset states $|A_{i s_i,s'_i}\rangle_i$ and construct $R_i^0=\mathsf{iO}(A_i+s_i)$ and $R_i^1=\mathsf{iO}(A_i+s'_i)$ for every $i\in[\ell_0]$.
   (c) Samples keys K_2,K_3 from the respective secondary PRFs and use $R_i^0=\mathsf{iO}(A_i+s_i)$ and $R_i^1=\mathsf{iO}(A_i+s'_i)$ along with $k_{x^{\mathcal{B}},x^{\mathcal{C}}}$ to construct P, as given in fig. 11.
4. Ch computes $y_1^{\mathcal{B}}=\mathsf{PRG}(\mathsf{PRF.Eval}(k,x^{\mathcal{B}}))$, $y_1^{\mathcal{C}}=\mathsf{PRG}(\mathsf{PRF.Eval}(k,x^{\mathcal{C}}))$ ~~Ch samples $u^{\mathcal{B}},u^{\mathcal{C}}\xleftarrow{\$}\{0,1\}^m$ and computes $y_1^{\mathcal{B}}=\mathsf{PRG}(u^{\mathcal{B}}),y_1^{\mathcal{C}}=\mathsf{PRG}(u^{\mathcal{C}})$~~ and uses $y_1^{\mathcal{B}},y_1^{\mathcal{C}}$ along with $R_i^0,R_i^1,\mathsf{iO}(P),\mathsf{PRG}$ to construct C as depicted in fig. 16.
5. Ch samples $y_1,y_2\xleftarrow{\$}\{0,1\}^m$, and computes $y_0^{\mathcal{B}}\leftarrow\mathsf{PRG}(y_1),y_0^{\mathcal{C}}\leftarrow\mathsf{PRG}(y_2)$.
6. Ch constructs the circuit $\mathsf{iO}(D_0),\mathsf{iO}(D_1)$ where D_0 and D_1 are as depicted in figs. 14 and 17, respectively.
7. Ch samples $b\xleftarrow{\$}\{0,1\}$ and sends $(\mathsf{iO}(C),\{|A_{i s_i,s'_i}\rangle\}_i,\mathsf{iO}(D_b))$ to $\mathcal{A}$.
8. $\mathcal{A}(\tilde{C},\{|A_{i s_i,s'_i}\rangle\}_i,\mathsf{iO}(D_b))$ outputs a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
9. Apply $(\mathcal{B}(x^{\mathcal{B}},\cdot)\otimes\mathcal{C}(x^{\mathcal{C}},\cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_{\mathbf{B}},b_{\mathbf{C}})$.
10. Output 1 if $b_{\mathbf{B}}=b_{\mathbf{C}}=b$.

Hybrid$_{11}$:

1. $\mathcal{A}$ sends a key $s\in\mathcal{K}_\lambda$ and functions $\mu_{\mathcal{B}}$ and $\mu_{\mathcal{C}}$ to Ch.
2. Ch samples $x^{\mathcal{B}},x^{\mathcal{C}}\xleftarrow{\$}\{0,1\}^n$.
3. Ch samples $k\leftarrow\mathsf{KeyGen}$, and does the following:
   (a) ~~Computes $k_{x^{\mathcal{B}},x^{\mathcal{C}}}\leftarrow\mathsf{PRF.Puncture}(k,\{x^{\mathcal{B}},x^{\mathcal{C}}\})$.~~
   (b) Samples ℓ_0 coset states $|A_{i s_i,s'_i}\rangle_i$ and construct $R_i^0=\mathsf{iO}(A_i+s_i)$ and $R_i^1=\mathsf{iO}(A_i+s'_i)$ for every $i\in[\ell_0]$.
   (c) Samples keys K_2,K_3 from the respective secondary PRFs and use $R_i^0=\mathsf{iO}(A_i+s_i)$ and $R_i^1=\mathsf{iO}(A_i+s'_i)$ along with ~~$k_{x^{\mathcal{B}},x^{\mathcal{C}}}$~~ k to construct P, as given in fig. 11.

4. Ch computes $y_1^{\mathcal{B}} = \mathsf{PRG}(\mathsf{PRF.Eval}(k, x^{\mathcal{B}}))$, $y_1^{\mathcal{C}} = \mathsf{PRG}(\mathsf{PRF.Eval}(k, x^{\mathcal{C}}))$ and uses $y_1^{\mathcal{B}}, y_1^{\mathcal{C}}$ along with $R_i^0, R_i^1, \mathsf{iO}(P), \mathsf{PRG}$ to construct C as depicted in fig. 16.
5. Ch samples $y_1, y_2 \xleftarrow{\$} \{0,1\}^m$, and computes $y_0^{\mathcal{B}} \leftarrow \mathsf{PRG}(y_1), y_0^{\mathcal{C}} \leftarrow \mathsf{PRG}(y_2)$.
6. Ch constructs the circuit $\mathsf{iO}(D_0), \mathsf{iO}(D_1)$ where D_0 and D_1 are as depicted in figs. 14 and 17, respectively.
7. Ch samples $b \xleftarrow{\$} \{0,1\}$ and sends $(\mathsf{iO}(C), \{|A_{is_i,s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ to $\mathcal{A}$.
8. $\mathcal{A}(\tilde{C}, \{|A_{is_i,s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ outputs a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
9. Apply $(\mathcal{B}(x^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(x^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_{\mathbf{B}}, b_{\mathbf{C}})$.
10. Output 1 if $b_{\mathbf{B}} = b_{\mathbf{C}} = b$.

<u>Hybrid_{12}</u>:

1. $\mathcal{A}$ sends a key $s \in \mathcal{K}_\lambda$ and functions $\mu_{\mathcal{B}}$ and $\mu_{\mathcal{C}}$ to Ch.
2. Ch samples $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$.
3. Ch samples $k \leftarrow \mathsf{KeyGen}$, and computes $\mathsf{iO}(P), |A_{is_i,s'_i}\rangle_i \leftarrow \mathsf{CLLZ.CopyProtect}(k)$.
4. Ch computes $y_1^{\mathcal{B}} = \mathsf{PRG}(\mathsf{PRF.Eval}(k, x^{\mathcal{B}})), y_1^{\mathcal{C}} = \mathsf{PRG}(\mathsf{PRF.Eval}(k, x^{\mathcal{C}}))$.
5. Ch constructs $C = \mathsf{PRG} \cdot \mathsf{iO}(P)$.
6. Ch samples $y_1, y_2 \xleftarrow{\$} \{0,1\}^m$, and computes $y_0^{\mathcal{B}} \leftarrow \mathsf{PRG}(y_1), y_0^{\mathcal{C}} \leftarrow \mathsf{PRG}(y_2)$.
7. Ch constructs the circuit $\mathsf{iO}(D_0), \mathsf{iO}(D_1)$ where D_0 and D_1 are as depicted in figs. 14 and 17, respectively.
8. Ch samples $b \xleftarrow{\$} \{0,1\}$ and sends $(\mathsf{iO}(C), \{|A_{is_i,s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ to $\mathcal{A}$.
9. $\mathcal{A}(\tilde{C}, \{|A_{is_i,s'_i}\rangle\}_i, \mathsf{iO}(D_b))$ outputs a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
10. Apply $(\mathcal{B}(x^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(x^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_{\mathbf{B}}, b_{\mathbf{C}})$.
11. Output 1 if $b_{\mathbf{B}} = b_{\mathbf{C}} = b$.

Next, we give a reduction from Hybrid_12 to the *preponed security* of the CLLZ copy-protection (for the PRFs with the required property having the key-generation algorithm KeyGen as mentioned above) to finish the proof.The reduction does the following.

1. $R_{\mathcal{A}}$ runs $\mathcal{A}$ to get a circuit f and g.
2. $R_{\mathcal{A}}$ on receiving the copy-protected PRF, $\mathsf{iO}(P), \{|A_{is_i,s'_i}\rangle\}_i$ and $u^{\mathcal{B}}, u^{\mathcal{C}}$, computes $y^{\mathcal{B}} \leftarrow \mathsf{PRG}(u^{\mathcal{B}})$ and $y^{\mathcal{C}} = \mathsf{PRG}(u^{\mathcal{C}})$, and creates the circuit $\tilde{C} \leftarrow \mathsf{iO}(C)$ where $C = \mathsf{PRG} \cdot \mathsf{iO}(P)$. $R_{\mathcal{A}}$ also creates $\mathsf{iO}(D)$ where D on input x, v, y runs C on x, v to get y' and outputs $\perp$ if $y' \neq y$ or $y' = \perp$, else if $y' \in \{y_0^{\mathcal{B}}, y^{\mathcal{C}}\}$ outputs $g(x)$, else it runs the circuit W_s to output $W_s(x)$. $R_{\mathcal{A}}$ runs $\mathcal{A}$ on $\rho_k, \mathsf{iO}(D)$ and gets an output $\sigma_{\mathcal{B},\mathcal{C}}$, it then sends the corresponding registers of $\sigma_{\mathcal{B},\mathcal{C}}$ to both $R_{\mathcal{B}}$ and $R_{\mathcal{C}}$.

3. $R_{\mathcal{B}}$ and $R_{\mathcal{C}}$ receive $x^{\mathcal{B}}$ and $x^{\mathcal{C}}$ from the challenger and run the adversaries $\mathcal{B}(x^{\mathcal{B}}, \cdot)$ and $\mathcal{C}(x^{\mathcal{C}}, \cdot)$ respectively on $\sigma_{\mathcal{B},\mathcal{C}}$, to get the outputs $b^{\mathcal{B}}$ and $b^{\mathcal{C}}$ respectively,. $R_{\mathcal{B}}$ and $R_{\mathcal{C}}$ output $1 - b^{\mathcal{B}}$ and $1 - b^{\mathcal{C}}$, respectively.

Finally, we prove the indistinguishability of the hybrids to finish the proof.

Indistinguishability of hybrids

Claim 35. *Assuming the security of* iO*, hybrids* Hybrid_0 *and* Hybrid_1 *are computationally indistinguishable.*

Proof of Claim 35. For any function f, let $\mathcal{I}_f$ denote the image of f. Since $\mathcal{I}_{\mathsf{PRG}}$ is a negligible fraction of $\{0,1\}^{2m}$ and $y_0^{\mathcal{B}}, y_0^{\mathcal{C}}$ were chosen uniformly at random, with overwhelming probability $y_0^{\mathcal{B}}, y_0^{\mathcal{C}} \notin \mathcal{I}_{\mathsf{PRG}}$ and hence not in $\mathcal{I}_C$. Therefore with overwhelming probability over the choice of $y_0^{\mathcal{B}}, y_0^{\mathcal{C}}$, any (x, v, y) that satisfies this check also satisfies $y \notin \{y_0^{\mathcal{B}}, y_0^{\mathcal{C}}\}$. Hence with overwhelming probability, if $y' = y \neq \perp$, the penultimate check (item 3 in fig. 14) will always fail, and therefore, D_0 will always output $W_s(x)$. Hence with overwhelming probability, D_0 has the same functionality in both the hybrids, and therefore by iO guarantees, the indistinguishability of the hybrids holds. □

Claim 36. *Assuming the pseudorandomness of* PRG*, hybrids* Hybrid_1 *and* Hybrid_2 *are computationally indistinguishable.*

Proof of Claim 35. The proof is immediate. □

Claim 37. *Assuming the security of* iO*, hybrids* Hybrid_2 *and* Hybrid_3 *are computationally indistinguishable.*

Proof of Claim 37. The modification did not change the functionality of D_1 in this hybrid compared to the previous hybrid by the definition of $W_{s,x^{\mathcal{B}},x^{\mathcal{C}},\mu_{\mathcal{B}},\mu_{\mathcal{C}}}$ and the $\mathsf{Puncture}$ algorithm associated with $\mathcal{W}$.Hence, the indistinguishability follows from the iO guarantees. □

Claim 38. *Assuming the security of* iO*, hybrids* Hybrid_3 *and* Hybrid_4 *are computationally indistinguishable.*

Proof of Claim 38. The indistinguishability follows by the iO guarantees and the claim that with overwhelming probability, the functionalities of $\mathsf{PRG} \cdot \mathsf{iO}(P)$ and C in this hybrid are the same. The proof of the claim is as follows.

In the proof of correctness [CLLZ21, Lemma 7.13] of the CLLZ copy-protection scheme, it was shown that the probability over the keys for the secondary pseudorandom functions, that $x^{\mathcal{B}}, x^{\mathcal{C}}$ are in the hidden triggers, is negligible. Hence, with overwhelming probability over the secondary pseudorandom function keys, $(x^{\mathcal{B}}, v)$ and $(x^{\mathcal{C}}, v)$ will not satisfy the trigger condition for P and therefore, not run in the hidden-trigger mode[8]. Hence with the same overwhelming probability, the functionality of P will not change even if we skip the hidden trigger check for $\{x^{\mathcal{B}}, x^{\mathcal{C}}\}$. Note that conditioned on the functionality does not change for P by skipping the check for $\{x^{\mathcal{B}}, x^{\mathcal{C}}\}$, the

[8]Note that this property depends only on the secondary keys k_2 and k_3. Since, over the hybrids, we only punctured the primary key and not the two secondary keys, the same correctness guarantee holds in this hybrid as in the unpunctured case of hybrid 0.

functionality of C in Hybrid_2 and Hybrid_3 are the same. Hence, with overwhelming probability, the functionality of C in Hybrid_3 is the same as that of $\mathsf{PRG} \cdot \mathsf{iO}(P)$. □

Claim 39. *Assuming the security of* iO*, hybrids* Hybrid_4 *and* Hybrid_5 *are computationally indistinguishable.*

Proof. The indistinguishability holds because P was hardcoded directly only in the circuit in the circuit C in the previous hybrid, and in C, we never use the key P to evaluate on $\{x^{\mathcal{B}}, x^{\mathcal{C}}\}$, and hence the functionality did not change even after we punctured the PRF key hardcoded inside P in Hybrid_5, due to the puncturing correctness of the PRF. Hence the indistinguishability follows from the iO guarantee since we did not change the functionality of C. □

Claim 40. *Assuming the security of the pseudorandom function family* PRF*, hybrids* Hybrid_5 *and* Hybrid_6 *are computationally indistinguishable.*

Proof. The proof is immediate. □

Claim 41. *Assuming the pseudorandomness of* PRG*, hybrids* Hybrid_6 *and* Hybrid_7 *are computationally indistinguishable.*

Proof. The proof is immediate. □

Claim 42. *Assuming the security of* iO*, hybrids* Hybrid_7 *and* Hybrid_8 *are computationally indistinguishable.*

Proof. We will show that the functionality of D_1 did not change across the hybrids Hybrid_7 and Hybrid_8 (see figs. 15 and 17), and hence indistinguishability of the hybrids follows from the iO guarantees. Note that since C in Hybrid_8 satisfies $C(x^{\mathcal{B}}, v^{\mathcal{B}}) = y^{\mathcal{B}}$ and $C(x^{\mathcal{C}}, v^{\mathcal{C}}) = y^{\mathcal{C}}$ $\forall v^{\mathcal{B}} \in V^{\mathcal{B}}$ and $v^{\mathcal{C}} \in V^{\mathcal{C}}$, where $V^{\mathcal{B}}$ (respectively, $V^{\mathcal{C}}$) is the set of all v such that $(x^{\mathcal{B}}, v)$ (respectively, $(x^{\mathcal{C}}, v)$) passes the coset check in the normal mode (see item 2), respectively. Moreover, the image of C restricted to $\mathcal{X}_C \setminus \big((x^{\mathcal{B}}, v^{\mathcal{B}}) \cup (x^{\mathcal{C}}, v^{\mathcal{C}})\big)$, i.e.,

$$\mathcal{I}_{C_{\mathcal{X}_C \setminus (x^{\mathcal{B}}, v^{\mathcal{B}}) \cup (x^{\mathcal{C}}, v^{\mathcal{C}})}} \subset \mathcal{I}_{\mathsf{PRG}(\{0,1\}^m)},$$

where m is the output length of the PRF family, $(x^{\mathcal{B}}, v^{\mathcal{B}})$ (respectively, $(x^{\mathcal{C}}, v^{\mathcal{C}})$) is the short hand notation for $\{(x^{\mathcal{B}}, v) \mid w \in V^{\mathcal{B}}\}$ (respectively, $\{(x^{\mathcal{C}}, v) \mid w \in V^{\mathcal{C}}\}$). Since $\mathcal{I}_{\mathcal{I}_{\mathsf{PRG}}}$ is a negligible fraction of $\{0,1\}^{2m}$, $\mathcal{I}_{C_{\mathcal{X}_C \setminus (x^{\mathcal{B}}, v^{\mathcal{B}}) \cup (x^{\mathcal{C}}, v^{\mathcal{C}})}}$ is also a negligible fraction of $\{0,1\}^{2m}$. Since $y_1^{\mathcal{B}}, y_1^{\mathcal{C}}$ are sampled uniformly at random independent of the set $\mathcal{I}_{C_{\mathcal{X}_C \setminus (x^{\mathcal{B}}, v^{\mathcal{B}}) \cup (x^{\mathcal{C}}, v^{\mathcal{C}})}}$, except with negligible probability,

$$y_1^{\mathcal{B}}, y_1^{\mathcal{C}} \notin \mathcal{I}_{C_{\mathcal{X}_C \setminus (x^{\mathcal{B}}, v^{\mathcal{B}}) \cup (x^{\mathcal{C}}, v^{\mathcal{C}})}}.$$

Note that we did not change the description of C after Hybrid_3, hence as noted in Hybrid_3,

$$C(x^{\mathcal{B}}, v) \in \{y_1^{\mathcal{B}}, \bot\}, \quad C(x^{\mathcal{C}}, v) \in \{y_1^{\mathcal{C}}, \bot\}.$$

Therefore, combining the last two statements, except with negligible probability, the preimage(s) of $y_1^{\mathcal{B}}$ are of the form $(x^{\mathcal{B}}, v)$, and the only non-$\bot$ image of $x^{\mathcal{B}}$ is $y_1^{\mathcal{B}}$, and similarly for $y_1^{\mathcal{C}}$ and $x^{\mathcal{C}}$. Hence except with negligible probability, the check that $y' = y \neq \bot$ and $y \in \{y_1^{\mathcal{B}}, y_1^{\mathcal{C}}\}$ is equivalent to $y' = y \neq \bot$ and $x \in \{x^{\mathcal{B}}, x^{\mathcal{C}}\}$. Therefore with overwhelming probability, the functionality of D_1 in Hybrid_7 (see fig. 15) and in Hybrid_8 (see fig. 17) are the same.

□

Claim 43. *Assuming the pseudorandomness of* PRG*, hybrids* Hybrid_8 *and* Hybrid_9 *are computationally indistinguishable.*

Proof. The proof is immediate. □

Claim 44. *Assuming the puncturing security of the pseudorandom function family* PRF*, hybrids* Hybrid_9 *and* Hybrid_{10} *are computationally indistinguishable.*

Proof. The proof is immediate. □

Claim 45. *Assuming the security of* iO*, hybrids* Hybrid_{10} *and* Hybrid_{11} *are computationally indistinguishable.*

Proof. The proof is the same as that of Claim 39. □

Claim 46. *Assuming the security of* iO*, hybrids* Hybrid_{11} *and* Hybrid_{12} *are computationally indistinguishable.*

Proof. The proof is the same as that of Claim 38. □

□

Proof of Lemma 34. The proof is the same as that of Lemma 33 upto minor adaptations and hence we omit the proof. □

6 Applications

We discuss the applications of unclonable puncturable obfuscation:

- We identify an interesting class of circuits and show that copy-protection for this class of functionalities exist. We show this in Section 6.2.
- We generalize the result from bullet 1 to obtain an approach to copy-protect certain family of cryptographic schemes. This is discussed in Section 6.3.
- We show how to copy-protect evasive functions in Section 6.5.
- We show how to construct public-key single-decryptor encryption from UPO in Section 6.4.

6.1 Notations for the applications

All the search-based applications (i.e., the security of which can be written as a cloning game with trivial success probability negligible) are with respect to independent challenge distribution. By the generic transformation in [AKL23], this implies the applications also achieve security with respect to arbitrarily correlated challenge distribution.
A function class $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ is said to have a keyed circuit implementation $\mathfrak{C} = \{\{C_k\}_{k\in\mathcal{K}_\lambda}\}_{\lambda\in\mathbb{N}}$ if for every function in $\mathcal{F}$, there is a circuit C_k in $\mathfrak{C}$ that implements f, i.e., the canonical map S_λ mapping a circuit C to its functionality when seen as a map $\mathfrak{C}_\lambda \mapsto \mathcal{F}_\lambda$, is surjective. In addition,

if there exists a distribution $\mathcal{D}_{\mathcal{F}}$ on $\mathcal{F}$, and an efficiently samplamable distribution $\mathcal{D}_{\mathcal{K}}$ on $\mathcal{K}$, such that,

$$\{S_\lambda(C_k)\}_{k \leftarrow \mathcal{D}_{\mathcal{K}}(1^\lambda)} \approx \{f\}_{f \leftarrow \mathcal{D}_{\mathcal{F}}(1^\lambda)},$$

then $(\mathcal{D}_{\mathcal{K}}, \mathfrak{C})$ is called a keyed circuit implementation of $(\mathcal{D}_{\mathcal{F}}, \mathcal{F})$.

Since any circuit class can be represented as a keyed circuit class using universal circuits, there is no loss of generality in our definition of keyed circuit implementation.

6.2 Copy-Protection for Puncturable Function Classes

We identify a class of circuits associated with a security property defined below. We later show that this class of circuits can be copy-protected.

Definition 47 (Puncturable Security)**.** *Let $\mathfrak{C} = \{\mathfrak{C}_\lambda\}_{\lambda \in \mathbb{N}}$ be a puncturable keyed circuit class (as defined in Section 3.1). Let* Puncture *be the puncturing algorithm and $\mathcal{K}$ be the key space associated with $\mathfrak{C}$.*

We say that $(\mathfrak{C}, \mathsf{Puncture})$ satisfies $\mathcal{D}_{\mathcal{K}}$-puncturable security, where $\mathcal{D}_{\mathcal{K}}$ is a distribution on $\mathcal{K}$, where n is the input length of the circuits in $\mathfrak{C}_\lambda$, if the following holds: for any quantum polynomial time adversary $\mathcal{A}$,

$$\mathsf{Pr}\left[y = C_k(x_1) \ : \ \begin{array}{c} k \leftarrow \mathcal{D}_{\mathcal{K}}(1^\lambda) \\ (x_1,x_2) \xleftarrow{\$} \{0,1\}^{2n} \\ G_{k^*} \leftarrow \mathsf{Puncture}(k,x_1,x_2) \\ y \leftarrow \mathcal{A}(x_1, G_{k^*}) \end{array} \right] \leq \frac{1}{2^m} + \mathsf{negl}(\lambda),$$

for some negligible function negl*. In the above expression, $C_k \in \mathfrak{C}_\lambda$ and n is the input length and m is the output length of C_k.*

Remark 48. *A possible objection to the definition could be the inclusion of x_2 in the definition. The sole purpose of including x_2 is to help in the proof.*

Remark 49. *We may abuse the notation and denote $\mathcal{D}_{\mathcal{K}}$ to be a distribution on $\mathfrak{C}$. Specifically, circuit C is sampled from $\mathcal{D}_{\mathcal{K}}(1^\lambda)$ as follows: first sample $k \leftarrow \mathcal{K}_\lambda$ and then set $C = C_k$.*

Theorem 50. *Suppose $\mathcal{F} = \mathcal{F}_{\lambda \lambda \in \mathbb{N}}$ be a function class equipped with a distribution $\mathcal{D}_{\mathcal{F}}$ such that there exists a keyed circuit implementation (see Section 6.1) $(\mathcal{D}_{\mathcal{K}}, \mathfrak{C})$ satisfying the following:*

1. *$\mathfrak{C}$ is a puncturable keyed circuit class associated with the puncturing algorithm* Puncture *and key space $\mathcal{K}$*

2. *$\mathfrak{C}$ satisfies $\mathcal{D}_{\mathcal{K}}$-puncturable security (Definition 47).*

Suppose $\mathsf{UPO} = (\mathsf{Obf}, \mathsf{Eval})$ is a secure unclonable puncturable obfuscation scheme for $\mathfrak{C}$ associated with distribution $\mathcal{D}_{\mathcal{X}}$, where $\mathcal{D}_{\mathcal{X}}$ is defined to be a uniform distribution.

Then there exists a copy-protection scheme $(\mathsf{CopyProtect}, \mathsf{Eval})$ for $\mathfrak{C}$ satisfying $(\mathcal{D}_{\mathcal{K}}, \mathcal{D}_{\mathcal{X}})$-anti-piracy, with respect to $\mathfrak{C}$ as the keyed circuit implementation of $\mathcal{F}$, and $(\mathcal{D}_{\mathcal{K}}, \mathfrak{C})$ as the keyed circuit implementation of $(\mathcal{D}_{\mathcal{F}}, \mathcal{F})$.

Proof. We define the algorithms $\mathsf{CP} = (\mathsf{CopyProtect}, \mathsf{Eval})$ as follows:

- $\mathsf{CopyProtect}(1^\lambda, C)$: on input $C \in \mathfrak{C}_\lambda$ with input length $n(\lambda)$, it outputs ρ_C, where $\rho_C \leftarrow \mathsf{UPO.Obf}(1^\lambda, C)$.
- $\mathsf{Eval}(\rho_C, x)$: on input ρ_C, input $x \in \{0,1\}^n$, it outputs the result of $\mathsf{UPO.Eval}(\rho_C, x)$.

The correctness of the copy-protection scheme follows from the correctness of UPO.

Next, we prove $(\mathcal{D}_\mathcal{K}, \mathcal{D}_\mathcal{X})$-anti-piracy with respect to the keyed circuit implementation $(\mathcal{D}_\mathcal{K}, \mathfrak{C})$ (see Appendix A.1). Let $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ be a non-local adversary in the anti-piracy experiment $\mathsf{CP.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}_\mathcal{K},\mathcal{D}_\mathcal{X}}\left(1^\lambda\right)$ defined in Figure 32. Consider the following adversary $(\mathcal{R}_\mathcal{A}, \mathcal{R}_\mathcal{B}, \mathcal{R}_\mathcal{C})$ in the UPO security experiment $\mathsf{UPO.Expt}^{(\mathcal{R}_\mathcal{A},\mathcal{R}_\mathcal{B},\mathcal{R}_\mathcal{C}),\mathcal{D}_\mathcal{X},\mathfrak{C}}\left(1^\lambda, \cdot\right)$ (Figure 2), defined as follows:

- $\mathcal{R}_\mathcal{A}$ samples $k \leftarrow \mathcal{D}_\mathcal{K}(1^\lambda)$, and sends k to the challenger Ch in the UPO security experiment.
- $\mathcal{R}_\mathcal{A}$ runs $\mathcal{A}$ on the received obfuscated state ρ from Ch to get a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$ on registers $\mathbf{B}$ and $\mathbf{C}$.
- $\mathcal{R}_\mathcal{A}$ sends register $\mathbf{B}$ and key k to $\mathcal{B}$. Similarly, $\mathcal{R}_\mathcal{A}$ sends register $\mathbf{C}$ and key k to $\mathcal{C}$.
- Ch generates $(x^\mathcal{B}, x^\mathcal{C}) \leftarrow \mathcal{D}_\mathcal{X}$.
- $\mathcal{R}_\mathcal{B}$ on receiving the challenge $x^\mathcal{B}$, runs $\mathcal{B}$ on $(k, \sigma_\mathcal{B}, x^\mathcal{B})$ to obtain $y^\mathcal{B}$. $\mathcal{R}_\mathcal{B}$ outputs 0 if and only if $y^\mathcal{B} = C_{k_\mathcal{B}}(x^\mathcal{B})$, otherwise outputs 1.
- $\mathcal{R}_\mathcal{C}$ receives the challenge $x^\mathcal{C}$ and does the same as $\mathcal{R}_\mathcal{B}$ but on $(k, \sigma_\mathcal{C}, x^\mathcal{C})$.

Define the following quantities:

- p^{CP}: probability that $(\mathcal{B}, \mathcal{C})$ simultaneously output $(C_k(x^\mathcal{B}), C_k(x^\mathcal{C}))$ in $\mathsf{CP.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}_\mathcal{K},\mathcal{D}_\mathcal{X}}\left(1^\lambda\right)$.
- For $b \in \{0,1\}$, p_b^{UPO}: probability that $(\mathcal{R}_\mathcal{B}, \mathcal{R}_\mathcal{C})$ simultaneously output b in $\mathsf{UPO.Expt}^{(\mathcal{R}_\mathcal{A},\mathcal{R}_\mathcal{B},\mathcal{R}_\mathcal{C}),\mathcal{D}_\mathcal{X},\mathfrak{C}}\left(1^\lambda, b\right)$.

In order to prove the security of CP, we have to upper bound p^{CP}. We have the following:

- From the description of $(\mathcal{R}_\mathcal{A}, \mathcal{R}_\mathcal{B}, \mathcal{R}_\mathcal{C})$, $p^{\mathsf{CP}} = p_0^{\mathsf{UPO}}$.
- From the security of UPO, we have that $\frac{1}{2}p_0^{\mathsf{UPO}} + \frac{1}{2}p_1^{\mathsf{UPO}} \leq \frac{1}{2} + \nu_1(\lambda)$ for some negligible function $\nu_1(\lambda)$.

Combining the two, we have:

$$\frac{1}{2}p^{\mathsf{CP}} + \frac{1}{2}p_1^{\mathsf{UPO}} \leq \frac{1}{2} + \nu_1(\lambda) \tag{2}$$

Claim 51. *Assuming $\mathcal{D}_\mathcal{K}$-puncturable security of $\mathfrak{C}$, there exists a negligible function $\nu_2(\lambda)$ such that $p_1^{\mathsf{UPO}} \geq 1 - \nu_2(\lambda)$.*

Proof. Define the following quantities. Let $q_1^{\mathcal{R}_\mathcal{B}}$ (respectively, $q_1^{\mathcal{R}_\mathcal{C}}$) be the probability that $\mathcal{R}_\mathcal{B}$ (respectively, $\mathcal{R}_\mathcal{C}$) outputs 0. Hence, $p_1^{\mathsf{UPO}} \geq 1 - q_1^{\mathcal{R}_\mathcal{B}} - q_1^{\mathcal{R}_\mathcal{C}}$. We prove that $q_1^{\mathcal{R}_\mathcal{B}} \leq \nu_3(\lambda)$, for some negligible function $\nu_3(\lambda)$ and symmetrically, it would follow that $q_1^{\mathcal{R}_\mathcal{C}} \leq \nu_4(\lambda)$.

Suppose $q_1^{\mathcal{R}_\mathcal{B}}$ is not negligible. We design an adversary $\mathcal{A}_{\mathsf{punc}}$ participating in the security experiment of Definition 47. Adversary $\mathcal{A}_{\mathsf{punc}}$ proceeds as follows:

- $\mathcal{A}_{\mathsf{punc}}$ on receiving (x_1, G_{k^*}), where $G_{k^*} \leftarrow \mathsf{Puncture}(k, x_1, x_2)$, generates $\rho \leftarrow \mathsf{Obf}(1^\lambda, G_{k^*})$.
- It then runs $\sigma_{\mathcal{BC}} \leftarrow \mathcal{R}_{\mathcal{A}}(\rho)$, where $\sigma_{\mathcal{BC}}$ is defined on two registers $\mathbf{B}$ and $\mathbf{C}$.
- Finally, it outputs the result of $\mathcal{R}_{\mathcal{B}}$ on the register $\mathbf{B}$ and x_1.

By the above description, the event that $\mathcal{A}_{\mathsf{punc}}$ wins exactly corresponds to the event that $\mathcal{R}_{\mathcal{B}}$ outputs 0. That is, the probability that $\mathcal{A}_{\mathsf{punc}}$ wins is $q_1^{\mathcal{R}_{\mathcal{B}}}$. Since $q_1^{\mathcal{R}_{\mathcal{B}}}$ is not negligible, it follows that $\mathcal{A}_{\mathsf{punc}}$ breaks the puncturable security of $\mathfrak{C}$ with non-negligible probability, a contradiction. Thus, $q_1^{\mathcal{R}_{\mathcal{B}}}$ is negligible and symmetrically, $q_1^{\mathcal{R}_{\mathcal{C}}}$ is negligible. □

From the above claim, we have:

$$\frac{1}{2}p^{\mathsf{CP}} + \frac{1}{2}p_1^{\mathsf{UPO}} \geq \frac{1}{2}p^{\mathsf{CP}} + \frac{1}{2} - \frac{1}{2}\nu_2(\lambda) \tag{3}$$

Combining Equation (2) and Equation (3), we have:

$$p^{\mathsf{CP}} \leq 2\nu_1(\lambda) + \nu_2(\lambda).$$

This proves the theorem. □

Instantiations. In the theorem below, we call a pseudorandom function to be a 2-point puncturable pseudorandom function if a pseudorandom function can be punctured at 2 points. Such a function family can be instantiated, for instance, from post-quantum one-way functions [BGI14, BW13]. We obtain the following corollary.

Corollary 52. *Let $\mathfrak{C}$ be a class of 2-point puncturable pseudorandom functions. Assuming the existence of unclonable puncturable obfuscation for $\mathfrak{C}$, there exists a copy-protection scheme for $\mathfrak{C}$.*

6.3 Copy-Protection for Puncturable Cryptographic Schemes

We generalize the approach in the previous section to capture puncturable cryptographic schemes, rather than just puncturable functionalities.

Syntax. A cryptographic primitive that is a tuple of probabilistic polynomial time algorithms $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Puncture}, \mathsf{Verify})$ such that

- $\mathsf{Gen}(1^\lambda)$: takes a security parameter and generates a secret key sk and a public auxiliary information aux. We will assume without loss of generality that $\mathsf{sk} \in \{0,1\}^\lambda$.
- $\mathsf{Eval}(\mathsf{sk}, x)$: takes a secret key sk and an input x and outputs a output string y. This is a deterministic algorithm.
- $\mathsf{Puncture}(\mathsf{sk}, x_1, x_2)$: takes a secret key sk and a set of inputs (x_1, x_2) and outputs a circuit $G_{\mathsf{sk}, x_1, x_2}$. This is a deterministic algorithm.
- $\mathsf{Verify}(\mathsf{sk}, \mathsf{aux}, x, y)$: takes a secret key sk, an auxiliary information aux, an input x and an output y and either accepts or rejects.

Definition 53 (Puncturable cryptographic schemes)**.** *A cryptographic scheme* $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Puncture}, \mathsf{Verify})$ *is a puncturable cryptographic scheme if it satisfies the following properties:*

- **Correctness***: The correctness property states that for any input* x*,* $\mathsf{Verify}(\mathsf{sk}, \mathsf{aux}, x, \mathsf{Eval}(x))$ *accepts, where* $(\mathsf{sk}, \mathsf{aux}) \leftarrow \mathsf{Gen}(1^\lambda)$*.*

- **Correctness of Punctured Circuit***: The correctness of punctured circuit states that for any set of inputs* $\{x_1, x_2\}$*, and* $G_{\mathsf{sk},x_1,x_2} \leftarrow \mathsf{Puncture}(\mathsf{sk}, x_1, x_2)$*, where* $(\mathsf{sk}, \mathsf{aux}) \leftarrow \mathsf{Gen}(1^\lambda)$*, it holds that* $G_{\mathsf{sk},x_1,x_2}(x) = \mathsf{Eval}(\mathsf{sk}, x)$ *for all* $x \notin \{x_1, x_2\}$ *and* $G_{\mathsf{sk},x_1,x_2}(x)$ *outputs* $\perp$ *if* $x \in \{x_1, x_2\}$*.*

- **Security***: We say that a puncturable cryptographic scheme* $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Puncture}, \mathsf{Verify})$ *satisfies puncturable security if the following holds: for any quantum polynomial time adversary* $\mathcal{A}$*,*

$$\Pr\left[\mathsf{Verify}(\mathsf{sk}, \mathsf{aux}, x_1, y) = 1 \ : \ \substack{(\mathsf{sk},\mathsf{aux})\leftarrow\mathsf{Gen}(1^\lambda) \\ x_1,x_2 \xleftarrow{\$} \{0,1\}^n \\ G_{\mathsf{sk},x_1,x_2}\leftarrow\mathsf{Puncture}(\mathsf{sk},x_1,x_2) \\ y\leftarrow\mathcal{A}(x_1,\mathsf{aux},G_{\mathsf{sk},x_1,x_2})}\right] \leq \mathsf{negl}(\lambda),$$

  for some negligible function negl*.*

Remark 54. *A possible objection to the definition could be the inclusion of* m_2 *in the definition. The sole purpose of including* m_2 *is to help in the proof. Assuming* iO *and length-doubling* PRG*, this added restriction does not rule out function classes further since, given* iO *and* PRG*, any function class that satisfies the above definition without the additional puncture point has a circuit representation that satisfies the puncturing security with this additional point of puncture* m_2*.*

<u>$\mathsf{PCS.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),}(1^\lambda)$:</u>

- Ch samples $\mathsf{sk}, \mathsf{aux} \leftarrow \mathsf{Gen}(1^\lambda)$, and generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \mathsf{Eval}(\mathsf{sk}, \cdot))$ and sends $(\rho_{\mathsf{sk}}, \mathsf{aux})$ to $\mathcal{A}$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- Ch samples $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$.
- Apply $(\mathcal{B}(x^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(x^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(y^{\mathcal{B}}, y^{\mathcal{C}})$.
- Output 1 if $\mathsf{Verify}(\mathsf{sk}, \mathsf{aux}, x^{\mathcal{B}}, y^{\mathcal{B}}) = 1$ and $\mathsf{Verify}(\mathsf{sk}, \mathsf{aux}, x^{\mathcal{C}}, y^{\mathcal{C}}) = 1$.

Figure 18: Anti-piracy experiment with uniform and independent challenge distribution:

Lemma 55. *Suppose* $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Puncture}, \mathsf{Verify})$ *is a puncturable cryptographic scheme. Let* UPO *be a unclonable puncturable obfuscation for the puncuturable keyed circuit class* $\{\mathfrak{C}_\lambda = \{\mathsf{Eval}(\mathsf{sk}, \cdot)\}_{\mathsf{sk}\in\{0,1\}^\lambda}$

parametrized by the secret keys, equipped with $\mathsf{Puncture}$ *as the puncturing algorithm. Then for every QPT adversary* $(\mathcal{A}, \mathcal{B}, \mathcal{C})$*, there exists a negligible function* negl *such that the following holds:*

$$\mathsf{Pr}\left[1 \leftarrow \mathsf{PCS.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C})}\left(1^{\lambda}\right)\right] \leq \mathsf{negl}(\lambda),$$

where $\mathsf{PCS.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C})}$ *is defined in Figure 18.*

Proof of lemma 55. Let $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ be a non-local adversary in the anti-piracy experiment $\mathsf{PCS.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C})}$ (Figure 19). Consider the following adversary $(\mathcal{R}_{\mathcal{A}}, \mathcal{R}_{\mathcal{B}}, \mathcal{R}_{\mathcal{C}})$ in the UPO security experiment $\mathsf{UPO.Expt}^{(\mathcal{R}_{\mathcal{A}},\mathcal{R}_{\mathcal{B}},\mathcal{R}_{\mathcal{C}}),\mathcal{D}_{\mathcal{X}},\mathfrak{C}}$ (Figure 2), defined as follows:

- $\mathcal{R}_{\mathcal{A}}$ samples $(\mathsf{sk}, \mathsf{aux}) \leftarrow \mathsf{Gen}(1^{\lambda})$, and sends sk to the challenger Ch in the UPO security experiment.
- $\mathcal{R}_{\mathcal{A}}$ receives ρ from Ch and runs $\mathcal{A}$ on (ρ, aux) from Ch to get a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- $\mathcal{R}_{\mathcal{A}}$ outputs $\mathsf{sk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{C}}, \mathsf{aux}_{\mathcal{B}}, \mathsf{aux}_{\mathcal{C}}, \sigma_{\mathcal{B},\mathcal{C}}$ where $\mathsf{sk}_{\mathcal{B}} = \mathsf{sk}_{\mathcal{C}} = \mathsf{sk}$ and $\mathsf{aux}_{\mathcal{B}} = \mathsf{aux}_{\mathcal{C}} = \mathsf{aux}$.
- $\mathcal{R}_{\mathcal{B}}$ receives the challenge $x^{\mathcal{B}}$ from Ch and $(\mathsf{sk}_{\mathcal{B}}, \mathsf{aux}_{\mathcal{B}}, \sigma_{\mathcal{B}})$ from $\mathcal{R}_{\mathcal{A}}$ and runs $\mathcal{B}$ on $\sigma_{\mathcal{B}}$ to obtain $y^{\mathcal{B}}$. $\mathcal{R}_{\mathcal{B}}$ outputs 0 if and only if $\mathsf{Verify}(\mathsf{sk}, \mathsf{aux}, x^{\mathcal{B}}, y^{\mathcal{B}}) = 1$, otherwise outputs 1.
- $\mathcal{R}_{\mathcal{C}}$ does the same but on $(\mathsf{aux}_{\mathcal{C}}, \sigma_{\mathcal{C}})$ and the challenge $x^{\mathcal{C}}$.

Note that the view of $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ in $\mathsf{Expt}^{(\mathcal{R}_{\mathcal{A}},\mathcal{R}_{\mathcal{B}},\mathcal{R}_{\mathcal{C}}),\mathcal{U}\times\mathcal{U},\mathfrak{C}}\left(1^{\lambda}, 0\right)$ is identical to the UPO experiment, and the event $1 \leftarrow \mathsf{Expt}^{(\mathcal{R}_{\mathcal{A}},\mathcal{R}_{\mathcal{B}},\mathcal{R}_{\mathcal{C}}),\mathcal{U}\times\mathcal{U},\mathfrak{C}}\left(1^{\lambda}, 0\right)$ corresponds to $1 \leftarrow \mathsf{Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),(\mathsf{Gen},\mathsf{Eval},\mathsf{Puncture},\mathsf{Verify}),\mathsf{UPO}}\left(1^{\lambda}\right)$. Let

$$p_b \equiv \mathsf{Pr}[1 \leftarrow \mathsf{Expt}^{(\mathcal{R}_{\mathcal{A}},\mathcal{R}_{\mathcal{B}},\mathcal{R}_{\mathcal{C}}),\mathcal{U}\times\mathcal{U},\mathfrak{C}}\left(1^{\lambda}, b\right)], \forall b \in \{0, 1\}.$$

Hence,

$$p_0 = \mathsf{Pr}[1 \leftarrow \mathsf{Expt}^{(\mathcal{R}_{\mathcal{A}},\mathcal{R}_{\mathcal{B}},\mathcal{R}_{\mathcal{C}}),\mathcal{U}\times\mathcal{U},\mathfrak{C}}\left(1^{\lambda}, 0\right)] \tag{4}$$

$$= \mathsf{Pr}\left[1 \leftarrow \mathsf{Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),(\mathsf{Gen},\mathsf{Eval},\mathsf{Puncture},\mathsf{Verify}),\mathsf{UPO}}\left(1^{\lambda}\right)\right]. \tag{5}$$

Therefore, it is enough to show that p_0 is negligible.

Note that by the UPO-security (see Definition 8) of the UPO scheme, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\mathsf{Pr}[b = 0]p_0 + \mathsf{Pr}[b = 1]p_1 = \frac{p_0 + p_1}{2} \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Hence,

$$p_0 \leq 1 + 2\mathsf{negl}(\lambda) - p_1. \tag{6}$$

Let $q_1^{\mathcal{R}_{\mathcal{B}}}$ (respectively, $q_1^{\mathcal{R}_{\mathcal{C}}}$) be the probability that $\mathcal{R}_{\mathcal{B}}$ ($\mathcal{R}_{\mathcal{C}}$) outputs 0, i.e., the inside adversary $\mathcal{B}$ (respectively, $\mathcal{C}$) passed verification, in the experiment $\mathsf{Expt}^{(\mathcal{R}_{\mathcal{A}},\mathcal{R}_{\mathcal{B}},\mathcal{R}_{\mathcal{C}}),\mathcal{U}\times\mathcal{U},\mathfrak{C}}\left(1^{\lambda}, 1\right)$.

Note that the event $0 \leftarrow \mathsf{Expt}^{(\mathcal{R}_{\mathcal{A}},\mathcal{R}_{\mathcal{B}},\mathcal{R}_{\mathcal{C}}),\mathcal{U}\times\mathcal{U},\mathfrak{C}}\left(1^{\lambda}, 1\right)$ corresponds to either $\mathcal{R}_{\mathcal{B}}$ outputs 0 or $\mathcal{R}_{\mathcal{C}}$ outputs 0 in $\mathsf{Expt}^{(\mathcal{R}_{\mathcal{A}},\mathcal{R}_{\mathcal{B}},\mathcal{R}_{\mathcal{C}}),\mathcal{U}\times\mathcal{U},\mathfrak{C}}\left(1^{\lambda}, 1\right)$. Hence,

$$\mathsf{Pr}\left[0 \leftarrow \mathsf{Expt}^{(\mathcal{R}_{\mathcal{A}},\mathcal{R}_{\mathcal{B}},\mathcal{R}_{\mathcal{C}}),\mathcal{U}\times\mathcal{U},\mathfrak{C}}\left(1^{\lambda}, 1\right)\right] \leq q_1^{\mathcal{R}_{\mathcal{B}}} + q_1^{\mathcal{R}_{\mathcal{C}}}.$$

Therefore,

$$p_1 = 1 - \mathsf{Pr}\left[0 \leftarrow \mathsf{Expt}^{(\mathcal{R}_\mathcal{A},\mathcal{R}_\mathcal{B},\mathcal{R}_\mathcal{C}),\mathcal{U}\times\mathcal{U},\mathfrak{C}}\left(1^\lambda, 1\right)\right] \geq 1 - q_1^{\mathcal{R}_\mathcal{B}} - q_1^{\mathcal{R}_\mathcal{C}}.$$

Combining with Equation (6), we conclude

$$p_0 \leq 1 + 2\mathsf{negl}(\lambda) - (1 - q_1^{\mathcal{R}_\mathcal{B}} - q_1^{\mathcal{R}_\mathcal{C}}) = q_1^{\mathcal{R}_\mathcal{B}} + q_1^{\mathcal{R}_\mathcal{C}} + 2\mathsf{negl}(\lambda). \quad (7)$$

Hence, it is enough to show that $q_1^{\mathcal{R}_\mathcal{C}}$ and $q_1^{\mathcal{R}_\mathcal{B}}$ are negligible.

Consider the adversary $A_{\mathcal{A},\mathcal{B}}$ in the puncturing security experiment given in Definition 56 for the puncturable signature scheme $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Puncture}, \mathsf{Verify})$.

- $A_{\mathcal{A},\mathcal{B}}$, on receiving $x_1, G_{\mathsf{sk},x_1,x_2}$ generates $\rho \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{Eval}(G_{\mathsf{sk},x_1,x_2}, \cdot))$.
- Then, runs $\sigma_{\mathcal{B},\mathcal{C}} \leftarrow \mathcal{A}(\rho)$.
- Finally, outputs $\mathcal{B}(\sigma_\mathcal{B})$.

It is easy to see that the event of $A_{\mathcal{A},\mathcal{B}}$ exactly corresponds with the event of $\mathcal{R}_\mathcal{B}$ outputting 1 in $\mathsf{Expt}^{(\mathcal{R}_\mathcal{A},\mathcal{R}_\mathcal{B},\mathcal{R}_\mathcal{C}),\mathcal{U}\times\mathcal{U},\mathfrak{C}}\left(1^\lambda, 1\right)$, where x_1 corresponds to $x^\mathcal{B}$. Therefore, by the puncturing security of $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Puncture}, \mathsf{Verify})$, there exists a negligible function $\epsilon_1(\lambda)$ such that,

$$q_1^\mathcal{B} = \mathsf{Pr}\left[\mathsf{Verify}(\mathsf{sk}, \mathsf{aux}, x_1, \mathsf{sig}) = 1 \;:\; \begin{array}{c} (\mathsf{sk},\mathsf{aux})\leftarrow\mathsf{Gen}(1^\lambda) \\ x_1,x_2 \xleftarrow{\$} \{0,1\}^n \\ G_{\mathsf{sk},x_1,x_2}\leftarrow\mathsf{Puncture}(\mathsf{sk},\{x_1,x_2\}) \\ \mathsf{sig}\leftarrow A_{\mathcal{A},\mathcal{B}}(x_1,\mathsf{aux},G_{\mathsf{sk},x_1,x_2}) \end{array}\right] \leq \epsilon_1.$$

Similarly, by considering the adversary $A_{\mathcal{A},\mathcal{C}}$ which is $A_{\mathcal{A},\mathcal{B}}$ with the $\mathcal{B}$ replaced as $\mathcal{C}$, we conclude that there exists a negligible function $\epsilon_2(\lambda)$ such that

$$q_1^\mathcal{C} = \mathsf{Pr}\left[\mathsf{Verify}(\mathsf{sk}, \mathsf{aux}, x_1, \mathsf{sig}) = 1 \;:\; \begin{array}{c} (\mathsf{sk},\mathsf{aux})\leftarrow\mathsf{Gen}(1^\lambda) \\ x_1,x_2 \xleftarrow{\$} \{0,1\}^n \\ G_{\mathsf{sk},x_1,x_2}\leftarrow\mathsf{Puncture}(\mathsf{sk},\{x_1,x_2\}) \\ \mathsf{sig}\leftarrow A_{\mathcal{A},\mathcal{C}}(x_1,\mathsf{aux},G_{\mathsf{sk},x_1,x_2}) \end{array}\right] \leq \epsilon_2.$$

Therefore, we conclude that both $q_1^{\mathcal{R}_\mathcal{C}}$ and $q_1^{\mathcal{R}_\mathcal{B}}$ are negligible in λ, which in combination with Equation (7) completes the proof of the anti-piracy.

□

6.3.1 Copy-Protection for Signatures

Definition 56 (Puncturable digital signatures [BSW16]). *Suppose* $\mathsf{DS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *be a digital signature with message length* $n = n(\lambda)$ *and signature length* $s = s(\lambda)$*. Let* $\mathsf{Puncture}, \mathsf{Sign}^*$ *be efficient polynomial time algorithms such that* $\mathsf{Puncture}()$ *takes as input a secret key and a message (or a polynomial number of messages)* (sk, m) *and outputs* sk_m*, and* Sign^* *is the signing algorithm for punctured keys such that* $\mathsf{Sign}^*(\mathsf{sk}_m, \cdot)$ *has the same functionality as* $\mathsf{Sign}^*(\mathsf{sk}_m, \cdot)$ *on all messagesm*$' \neq m$ *and* $\mathsf{Sign}^*(\mathsf{sk}_m, m')$ *outputs* $\perp$.

We say that a puncturable digital signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Puncture}, \mathsf{Verify}, \mathsf{Sign}^*)$ *satisfies puncturable security if the following holds: for any quantum polynomial time adversary* $\mathcal{A}$,

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, x_1, \mathsf{sig}) = 1 \ : \ \begin{array}{c} (\mathsf{sk},\mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ m_1, m_2 \xleftarrow{\$} \{0,1\}^n \\ \mathsf{sk}_{m_1,m_2} \leftarrow \mathsf{Puncture}(\mathsf{sk}, \{m_1, m_2\}) \\ \mathsf{sig} \leftarrow \mathcal{A}(m_1, \mathsf{vk}, \mathsf{sk}_{m_1,m_2}) \end{array}\right] \leq \mathsf{negl}(\lambda),$$

for some negligible function $\mathsf{negl}()$.

Remark 57. *A possible objection to the definition could be the inclusion of* m_2 *in the definition. The sole purpose of including* m_2 *is to help in the proof. Assuming* iO *and length-doubling* PRG*, this added restriction does not rule out function classes further since, given* iO *and* PRG*, it can be shown that any function class that satisfies the above definition without the additional puncture point has a circuit representation that satisfies the puncturing security with this additional point of puncture* m_2.

Definition 58 (Adapted from [LLQZ22])**.** *A copy-protection Scheme for a signature scheme with message length* $n(\lambda$ *and signature length* $s(\lambda)$ *consists of the following algorithms:*

- $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda)$: *on input a security parameter* 1^λ*, returns a classical secret key* sk *and a classical verification key* vk*.*
- $\rho_{\mathsf{sk}} \leftarrow \mathsf{QKeyGen}(\mathsf{sk})$: *takes a classical secret key* sk *and outputs a quantum signing key* ρ_{sk}*.*
- $\mathsf{sig} \leftarrow \mathsf{Sign}(\rho_{\mathsf{sk}}, m)$: *takes a quantum signing key* ρ_{sk} *and a message* m *for* $m \in \{0,1\}^{n(\lambda)}$*, and outputs a classical signature* sig*.*
- $b \leftarrow \mathsf{Verify}(\mathsf{vk}, m, \mathsf{sig})$ *takes a classical verification key* vk*, a message* m *and a classical signature* sig*, and outputs a bit* b *indicating accept* ($b = 1$) *or reject* ($b = 0$)*.*

Correctness *For every message* $m \in \{0,1\}^{n(\lambda)}$*, there exists a negligible function* $\delta(\lambda)$*, (also called the correctness precision) such that*

$$\Pr[\mathsf{sk}, \mathsf{vk} \leftarrow \mathsf{Gen}(\lambda); \rho_{\mathsf{sk}} \leftarrow \mathsf{QKeyGen}(\mathsf{sk}), \mathsf{sig} \leftarrow \mathsf{Sign}(\rho_{\mathsf{sk}}, m) : \mathsf{Verify}(\mathsf{vk}, \mathsf{sig}) = 1] \geq 1 - \delta(\lambda).$$

Security *We say that a copy-protection scheme for signatures* $\mathsf{CP\text{-}DS} = (\mathsf{Gen}, \mathsf{QKeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *satisfies anti-piracy with respect to the product distribution* $\mathcal{U} \otimes \mathcal{U}$ *if for every efficient adversary* $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ *in Experiment 19 there exists a negligible function* $\mathsf{negl}()$ *such that*

$$\Pr\left[1 \leftarrow \mathsf{Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathsf{CP\text{-}DS}}\left(1^\lambda\right)\right] \leq \mathsf{negl}(\lambda).$$

Theorem 59. *Suppose* $\mathsf{DS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Puncture}, \mathsf{Verify}, \mathsf{Sign}^*)$ *be a puncturable digital signature with signa length* $n(\lambda)$ *and signature length* $s(\lambda)$*. Let* $\mathsf{Puncture}'$ *be an algorithm derived from* DS.Puncture *similar to how* Puncture *is derived from* Puncture *in Lemma 55*

Given a unclonable puncturable obfuscation scheme (Obf, Eval) *with* UPO*-security (see Definition 8) for* $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ *where* $\mathcal{F}_\lambda = \{\mathsf{Sign}(k, \cdot)\}_{k \in Support(\mathsf{Gen}(1^\lambda))}$*, equipped with* Puncture *as the puncturing algorithm, with respect to* $\mathcal{D}_\mathcal{X} = \mathcal{U} \times \mathcal{U}$*, there exists a copy-protection for signature scheme* (CopyProtect, Eval) *where* Gen, Verify *are the same as the puncturable signature scheme and* $\mathsf{QKeyGen}(\mathsf{sk}) = \mathsf{Obf}(\mathsf{Sign}(\mathsf{sk}, \cdot))$ *and the* Sign() *algorithm is the same as the* Eval() *algorithm of the* UPO *scheme.*

$\underline{\mathsf{Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathsf{CP\text{-}DS}}\left(1^\lambda\right)}$:

- Ch samples $\mathsf{sk},\mathsf{vk} \leftarrow \mathsf{Gen}(1^\lambda)$ and generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{QKeyGen}(\mathsf{sk})$ and sends $(\rho_{\mathsf{sk}},\mathsf{vk})$ to $\mathcal{A}$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- Ch samples $m^{\mathcal{B}}, m^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$.
- Apply $(\mathcal{B}(m^{\mathcal{B}},\cdot)\otimes\mathcal{C}(m^{\mathcal{C}},\cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(\mathsf{sig}^{\mathcal{B}},\mathsf{sig}^{\mathcal{C}})$.
- Output 1 if $\mathsf{Verify}(\mathsf{vk},m^{\mathcal{B}},\mathsf{sig}^{\mathcal{B}})=1$ and $\mathsf{Verify}(\mathsf{vk},m^{\mathcal{C}},\mathsf{sig}^{\mathcal{C}})=1$.

Figure 19: Anti-piracy experiment with uniform and independent challenge distribution for copy-protection of signatures.

Proof of Theorem 59. The correctness of the copy-protection of signatures scheme directly follows from the UPO-correctness guarantees, see Section 3. Next, we prove anti-piracy. Let $(\mathcal{A},\mathcal{B},\mathcal{C})$ be a non-local adversary in the anti-piracy experiment $\mathsf{Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathsf{CP\text{-}DS}}\left(1^\lambda\right)$ given in Figure 19. By the puncturing security and correctness of $\mathsf{DS}=(\mathsf{Gen},\mathsf{Sign},\mathsf{Puncture},\mathsf{Verify},\mathsf{Sign}^*)$, $(\mathsf{Gen},\mathsf{Sign},\mathsf{Puncture}',\mathsf{Verify}')$ is a puncturable cryptographic scheme where vk is the auxiliary information aux, the message space is the input space, the signature is the output space, $\mathsf{Gen}=\mathsf{DS.Gen}$, $\mathsf{Eval}=\mathsf{DS.Sign}$, $\mathsf{Verify}'(\mathsf{sk},\mathsf{vk},m,\mathsf{sig})=\mathsf{DS.Verify}(\mathsf{vk},m,\mathsf{sig})$ and $\mathsf{Puncture}'$ is the efficient algorithm that takes a key sk and a set of input points $\{x_1,x_2\}$, generates $\mathsf{sk}_{x_1,x_2}\leftarrow\mathsf{DS.Puncture}(\mathsf{sk},x_1,x_2)$ and outputs $\mathsf{Sign}^*(\mathsf{sk}_{x_1,x_2},\cdot)$.

Therefore, by Lemma 55, for any adversary $(\mathcal{A},\mathcal{B},\mathcal{C})$ in the anti-piracy experiment $\mathsf{Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),(\mathsf{Gen},\mathsf{Sign},\mathsf{Puncture},\mathsf{Verify}),\mathsf{UPO}}\left(1^\lambda\right)$, there exists a negligible function $\mathsf{negl}()$ such that,

$$\Pr\left[1\leftarrow\mathsf{Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),(\mathsf{Gen},\mathsf{Sign},\mathsf{Puncture},\mathsf{Verify}),\mathsf{UPO}}\left(1^\lambda\right)\right]\leq\mathsf{negl}(\lambda).$$

However, $\mathsf{Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),(\mathsf{Gen},\mathsf{Sign},\mathsf{Puncture},\mathsf{Verify}),\mathsf{UPO}}\left(1^\lambda\right)$ and $\mathsf{Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathsf{CP\text{-}DS}}\left(1^\lambda\right)$ are the same experiments and therefore, we conclude that anti-piracy holds for the CP-DS with respect to uniform and independent challenge distribution.

□

Remark 60. *By the same arguments as in the proof of theorem 59, it can be shown that any unclonable puncturable obfuscation scheme* $(\mathsf{Obf},\mathsf{Eval})$ *with* $\mathsf{Id}_{\mathcal{U}}$-$\mathsf{UPO}$ *security (see Definition 8) for any puncturable keyed circuit class in* $\mathsf{P/poly}$ *(see Section 3.1.1), is also a copy-protection scheme* $(\mathsf{CopyProtect},\mathsf{Eval})$ *for* $\mathcal{F}=\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ *with uniform and identical challenge distribution, where* $\mathsf{CopyProtect}()=\mathsf{Obf}()$.

Since copy-protection for signatures implies public-key quantum money schemes, we get the following corollary.

Corollary 61. *Suppose* $\mathsf{DS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Puncture}, \mathsf{Verify}, \mathsf{Sign}^*)$ *be a puncturable digital signature with signa length* $n(\lambda)$ *and signature length* $s(\lambda)$*. Let* $\mathsf{Puncture}'$ *be an algorithm derived from* $\mathsf{DS.Puncture}$ *similar to how* $\mathsf{Puncture}$ *is derived from* $\mathsf{Puncture}$ *in Lemma 55*

Given a unclonable puncturable obfuscation scheme $(\mathsf{Obf}, \mathsf{Eval})$ *with* UPO*-security (see Definition 8) for* $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ *where* $\mathcal{F}_\lambda = \{\mathsf{Sign}(k,\cdot)\}_{k \in Support(\mathsf{Gen}(1^\lambda))}$*, equipped with* $\mathsf{Puncture}$ *as the puncturing algorithm, with respect to* $\mathcal{D}_\mathcal{X} = \mathcal{U} \times \mathcal{U}$*, there exists a public-key quantum money scheme.*

6.4 Public-key Single-Decryptor Encryption

Construction Our construction is based on copy-protecting the decryption functionality of the Sahai-Waters public-key encryption scheme based on iO, PRF (mapping $n(\lambda)$ bits to $n(\lambda)$ bits), and PRG (mapping $\frac{n(\lambda)}{2}$ bits to $n(\lambda)$ bits). We assume a unclonable puncturable obfuscation scheme $\mathsf{UPO} = (\mathsf{Obf}, \mathsf{Eval})$ satisfying $\mathcal{U}$-generalized security (see Definition 9) for any generalized puncturable keyed circuit class in $\mathsf{P/poly}$. In the security proofs, we will be considering the circuit class $\mathfrak{C} = \{\{\mathsf{PRF.Eval}(k,\cdot)\}_{k \in \mathsf{Supp}(\mathsf{KeyGen}(1^\lambda))}\}_\lambda$ equipped with the distribution $\mathsf{PRF.Gen}(1^\lambda)$ on the PRF keys, and a puncturing or a generalized puncturing algorithms, derived accordingly from the $\mathsf{PRF.Puncture}$ algorithm.

Assumes: PRF family $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Puncture})$, length-doubling PRG, iO, UPO scheme $(\mathsf{Obf}, \mathsf{Eval})$.

$\mathsf{Gen}(1^\lambda)$

1. Sample a key $k \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.
2. Generate the circuit C that on input $r \leftarrow \{0,1\}^{\frac{n(\lambda)}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k, \mathsf{PRG}(r)) \oplus m)$.
3. Compute $\tilde{C} \leftarrow \mathsf{iO}(C)$.
4. Output $(\mathsf{sk}, \mathsf{pk}) = (k, \tilde{C})$.

$\mathsf{QKeyGen}(\mathsf{sk})$

1. Compute $\tilde{F} \leftarrow \mathsf{iO}(\mathsf{PRF.Eval}(\mathsf{sk}, \cdot))$.
2. Output $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{F})$[9].

$\mathsf{Enc}(\mathsf{pk}, m)$

1. Interprete $\mathsf{pk} = \tilde{C}$
2. Sample $r \xleftarrow{\$} \{0,1\}^{\frac{n}{2}}$.
3. Output $\mathsf{ct} = \tilde{C}(r, m)$.

$\mathsf{Dec}(\rho_{\mathsf{sk}}, \mathsf{ct})$

1. Interprete $\mathsf{ct} = y, z$.
2. Output $m = \mathsf{UPO.Eval}(\rho_{\mathsf{sk}}, y) \oplus z$.

Figure 20: A construction of single decryptor encryption based on [SW14] public-key encryption.

[9]We assume that it is possible to read off the security parameter from the secret key sk. For example, the secret key could start with 1^λ followed by a special symbol, and then followed by the actual key.

Theorem 62. *Assuming an indistinguishability obfuscation scheme* iO *for* $\mathsf{P/poly}$*, a puncturable pseudorandom function family* $\mathsf{PRF} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Puncture})$ *and a generalized unclonable puncturable obfuscation* UPO *for any generalized puncturable keyed circuit class in* $\mathsf{P/poly}$ *with respect to* $\mathcal{D}_{\mathcal{X}} = U \times U$*, there exists a* single decryptor encryption *scheme given in Figure 20 that satisfies correctness, search anti-piracy with independent and uniform distribution and* $\mathcal{D}_{\mathsf{iden\text{-}bit,ind\text{-}msg}}$*-selective* CPA*-style anti-piracy (see Appendix A.2).*

Proof. The proof follows by combining Lemma 63 and Propositions 64 and 66. □

Lemma 63. *The* single decryptor encryption *construction given in Figure 20 satisfies correctness with the same correctness precision as the underlying* UPO *scheme.*

The proof is immediate, so we omit the proof.

Proposition 64. *The* single decryptor encryption *construction given in Figure 20 satisfies search anti-piracy with independent and uniform distribution (see Appendix A.2) if the underlying* UPO *scheme satisfies unclonable puncturable obfuscation security for any puncturable keyed circuit class in* $\mathsf{P/poly}$.

We first identify a scheme $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Verify}, \mathsf{Puncture})$ (defined in Figure 21) based on the public-key encryption scheme given in [SW14], and show that it is a puncturable cryptographic scheme, as defined in Definition 53, see Lemma 65. This result would be required in the proof of Proposition 64 given on Page 64.

Assumes: PRF family $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Puncture})$, length-doubling PRG, iO, UPO scheme $(\mathsf{Obf}, \mathsf{Eval})$

$\mathsf{Gen}(1^\lambda)$: Generate $(k, \tilde{C}) \leftarrow \mathsf{SDE.Gen}(1^\lambda)$ where SDE is the single decryptor encryption given in Figure 20, and output $(\mathsf{sk}, \mathsf{aux})$ where $\mathsf{sk} = k$ and $\mathsf{aux} = \mathsf{pk}$.

$\mathsf{Eval}(\mathsf{sk}, x)$: Same as $\mathsf{PRF.Eval}(\mathsf{sk}, x)$.

$\mathsf{Verify}(\mathsf{sk}, \mathsf{aux}, x, y)$: Check if $\mathsf{PRF.Eval}(\mathsf{sk}, x) = y$ and if true outputs 1 else 0.

$\mathsf{Puncture}(\mathsf{sk}, x_1, x_2)$: Generate $\mathsf{sk}_{x_1,x_2} \leftarrow \mathsf{PRF.Puncture}(\mathsf{sk}, x_1, x_2)$ and output $\mathsf{PRF.Eval}(\mathsf{sk}_{x_1,x_2}, \cdot)$.

Figure 21: A construction of puncturable cryptographic scheme based on [SW14] public-key encryption.

Lemma 65. *The scheme* $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Puncture}, \mathsf{Verify})$ *given in Figure 21 is a puncturable cryptographic scheme, as defined in Definition 53.*

Proof. The correctness and correctness of punctured circuit for $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Puncture}, \mathsf{Verify})$ is immediate. Next, we prove the puncturable security.

Let A be an adversary in the puncturing experiment given in Definition 53 for the puncturable cryptographic scheme $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Puncture}, \mathsf{Verify})$. Hybrid_0:
Same as the puncturing security experiment given in Definition 53.

- Ch samples $k \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.
- Ch generates the circuit $\tilde{C} \leftarrow \mathsf{iO}(C)$ where C has k hardcoded and on input $r \leftarrow \{0,1\}^{\frac{n}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k, \mathsf{PRG}(r)) \oplus m)$.
- Ch samples $x_1, x_2 \xleftarrow{\$} \{0,1\}^n$.
- Ch generates $k_{x_1,x_2} \leftarrow \mathsf{PRF.Puncture}(k, \{x_1, x_2\})$.
- Ch sends $(x_1, k_{x_1,x_2}, \tilde{C})$ to A and gets back y.
- Ch computes $y_1 \leftarrow \mathsf{PRF.Eval}(k, x_1)$.
- Output 1 if $y = y_1$.

Hybrid_1:

- Ch samples $k \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.
- Ch generates the circuit $\tilde{C} \leftarrow \mathsf{iO}(C)$ where C has ~~k~~$k_{x^{\mathcal{B}},x^{\mathcal{C}}}$ hardcoded and on input $r \leftarrow \{0,1\}^{\frac{n}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}($~~$k$~~$k_{x^{\mathcal{B}},x^{\mathcal{C}}}, \mathsf{PRG}(r)) \oplus m)$.
- Ch samples $x_1, x_2 \xleftarrow{\$} \{0,1\}^n$.
- Ch generates $k_{x_1,x_2} \leftarrow \mathsf{PRF.Puncture}(k, \{x_1, x_2\})$.
- Ch sends $(x_1, k_{x_1,x_2}, \tilde{C})$ to A and gets back y.
- Ch computes $y_1 \leftarrow \mathsf{PRF.Eval}(k, x_1)$.
- Output 1 if $y = y_1$.

The proof of indistinguishability between Hybrid_0 and Hybrid_1 is as follows. Note that $x_1, x_2 \xleftarrow{\$} \{0,1\}^n$ and $\mathsf{Supp}(\mathsf{PRG}) \subset \{0,1\}^n$ has size $2^{\frac{n}{2}}$, and hence is a negligible fraction of $\{0,1\}^n$. Hence, with overwhelming probability $x_1, x_2 \notin \mathsf{Supp}(\mathsf{PRG})$. Therefore with overwhelming probability, C as in Hybrid_0 never computes $\mathsf{PRF.Eval}(k, \cdot)$ on x_1 or x_2 on any input query. Hence, replacing k with k_{x_1,x_2} inside C does not change the functionality of C, by the puncturing correctness of PRF. Therefore, indistinguishability holds by the iO guarantee.

Hybrid_2:

- Ch samples $k \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.
- Ch generates the circuit $\tilde{C} \leftarrow \mathsf{iO}(C)$ where C has ~~k~~$k_{x^{\mathcal{B}},x^{\mathcal{C}}}$ hardcoded and on input $r \leftarrow \{0,1\}^{\frac{n}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}($~~$k$~~$k_{x^{\mathcal{B}},x^{\mathcal{C}}}, \mathsf{PRG}(r)) \oplus m)$.
- Ch samples $x_1, x_2 \xleftarrow{\$} \{0,1\}^n$.

- Ch generates $k_{x_1,x_2} \leftarrow \mathsf{PRF.Puncture}(k, \{x_1, x_2\})$.
- Ch sends $(x_1, k_{x_1,x_2}, \tilde{C})$ to A and gets back y.
- Ch ~~computes $y_1 \leftarrow \mathsf{PRF.Eval}(k, x_1)$~~ samples $y_1 \xleftarrow{\$} \{0,1\}^n$.
- Output 1 if $y = y_1$.

The indistinguishability holds because the view of A in Hybrid_1 depends only on k_{x_1,x_2} and not on k. Hence, A cannot distinguish between $y_1 \leftarrow \mathsf{PRF.Eval}(k, x_1)$ with $y_1 \xleftarrow{\$} \{0,1\}^n$. Therefore, checking if y, the response of A is equal to y_1 when $y_1 \leftarrow \mathsf{PRF.Eval}(k, x_1)$ should be indistinguishable from the same experiment but with $y_1 \xleftarrow{\$} \{0,1\}^n$.

Finally, we argue that since y_1 is sampled independent of y, the probability that $y = y_1$, i.e., the output of Hybrid_2 is 1, is exactly $\frac{1}{2^n}$, which is a negligible function of λ since $n(\lambda) \in \mathrm{poly}(\lambda)$. □

Proof of Proposition 64. Let $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ be any adversary in $\mathsf{Search.SDE.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}}(1^\lambda)$ (see Figure 33). We will do a sequence of hybrids starting from the original anti-piracy experiment $\mathsf{Search.SDE.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}}(1^\lambda)$ for the single decryptor encryption scheme given in Figure 20. The changes are marked in blue.

Hybrid_0:
Same as $\mathsf{Search.SDE.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}}(1^\lambda)$ given in Figure 33 for the single decryptor encryption scheme in Figure 20.

- Ch samples $k \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.
- Ch samples $r^\mathcal{B}, r^\mathcal{C} \xleftarrow{\$} \{0,1\}^{\frac{n}{2}}$ and generates $x^\mathcal{B} \leftarrow \mathsf{PRG}(r^\mathcal{B})$ and $x^\mathcal{C} \leftarrow \mathsf{PRG}(r^\mathcal{C})$.
- Ch generates the circuit $\tilde{C} \leftarrow \mathsf{iO}(C)$ where C has k hardcoded and on input $r \leftarrow \{0,1\}^{\frac{n}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k, \mathsf{PRG}(r)) \oplus m)$.
- Ch generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{F})$ where $\tilde{F} \leftarrow \mathsf{iO}(\mathsf{PRF.Eval}(k, \cdot))$ and sends $(\rho_{\mathsf{sk}}, \tilde{C})$ to $\mathcal{A}$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- Ch samples $m^\mathcal{B}, m^\mathcal{C} \xleftarrow{\$} \{0,1\}^n$.
- Ch computes $\mathsf{ct}^\mathcal{B} = (x^\mathcal{B}, z^\mathcal{B})$ and $\mathsf{ct}^\mathcal{C} = (x^\mathcal{C}, z^\mathcal{C})$ where $z^\mathcal{B} = \mathsf{PRF.Eval}(k, x^\mathcal{B}) \oplus m^\mathcal{B}$ and $z^\mathcal{C} = \mathsf{PRF.Eval}(k, x^\mathcal{C}) \oplus m^\mathcal{C}$.
- Apply $(\mathcal{B}(\mathsf{ct}^\mathcal{B}, \cdot) \otimes \mathcal{C}(\mathsf{ct}^\mathcal{C}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(y^\mathcal{B}, y^\mathcal{C})$.
- Output 1 if $y^\mathcal{B} = m^\mathcal{B}$ and $y^\mathcal{C} = m^\mathcal{C}$.

Hybrid_1:

- Ch samples $k \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.

- Ch samples ~~$r^{\mathcal{B}}, r^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^{\frac{n}{2}}$ and generates $x^{\mathcal{B}} \leftarrow \mathsf{PRG}(r^{\mathcal{B}})$ and $x^{\mathcal{C}} \leftarrow \mathsf{PRG}(r^{\mathcal{C}})$~~ $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$.

- Ch generates the circuit $\tilde{C} \leftarrow \mathsf{iO}(C)$ where C has k hardcoded and on input $r \leftarrow \{0,1\}^{\frac{n}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k, \mathsf{PRG}(r)) \oplus m)$.

- Ch generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{F})$ where $\tilde{F} \leftarrow \mathsf{iO}(\mathsf{PRF.Eval}(k, \cdot))$ and sends $(\rho_{\mathsf{sk}}, \tilde{C})$ to $\mathcal{A}$.

- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.

- Ch samples $m^{\mathcal{B}}, m^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$.

- Ch computes $\mathsf{ct}^{\mathcal{B}} = (x^{\mathcal{B}}, z^{\mathcal{B}})$ and $\mathsf{ct}^{\mathcal{C}} = (x^{\mathcal{C}}, z^{\mathcal{C}})$ where $z^{\mathcal{B}} = \mathsf{PRF.Eval}(k, x^{\mathcal{B}}) \oplus m^{\mathcal{B}}$ and $z^{\mathcal{C}} = \mathsf{PRF.Eval}(k, x^{\mathcal{C}}) \oplus m^{\mathcal{C}}$.

- Apply $(\mathcal{B}(\mathsf{ct}^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(y^{\mathcal{B}}, y^{\mathcal{C}})$.

- Output 1 if $y^{\mathcal{B}} = m^{\mathcal{B}}$ and $y^{\mathcal{C}} = m^{\mathcal{C}}$.

The indistinguishability between Hybrid_0 and Hybrid_1 follows from the pseudorandomness of PRG.
Hybrid_2:

- Ch samples $k \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.

- Ch samples $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$.

- Ch generates the circuit $\tilde{C} \leftarrow \mathsf{iO}(C)$ where C has k hardcoded and on input $r \leftarrow \{0,1\}^{\frac{n}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k, \mathsf{PRG}(r)) \oplus m)$.

- Ch generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{F})$ where $\tilde{F} \leftarrow \mathsf{iO}(\mathsf{PRF.Eval}(k, \cdot))$ and sends $(\rho_{\mathsf{sk}}, \tilde{C})$ to $\mathcal{A}$.

- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.

- Ch samples ~~$m^{\mathcal{B}}, m^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$~~ $z^{\mathcal{B}}, z^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$ and computes $m^{\mathcal{B}} = \mathsf{PRF.Eval}(k, x^{\mathcal{B}}) \oplus z^{\mathcal{B}}$, $m^{\mathcal{C}} = \mathsf{PRF.Eval}(k, x^{\mathcal{C}}) \oplus z^{\mathcal{C}}$.

- Ch computes $\mathsf{ct}^{\mathcal{B}} = (x^{\mathcal{B}}, z^{\mathcal{B}})$ and $\mathsf{ct}^{\mathcal{C}} = (x^{\mathcal{C}}, z^{\mathcal{C}})$ ~~where $z^{\mathcal{B}} = \mathsf{PRF.Eval}(k, x^{\mathcal{B}}) \oplus m^{\mathcal{B}}$ and $z^{\mathcal{C}} = \mathsf{PRF.Eval}(k, x^{\mathcal{C}}) \oplus m^{\mathcal{C}}$~~.

- Apply $(\mathcal{B}(\mathsf{ct}^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(y^{\mathcal{B}}, y^{\mathcal{C}})$.

- Output 1 if $y^{\mathcal{B}} = m^{\mathcal{B}}$ and $y^{\mathcal{C}} = m^{\mathcal{C}}$.

The overall distribution on $(m^{\mathcal{B}}, z^{\mathcal{B}})$ and $(m^{\mathcal{C}}, z^{\mathcal{C}})$ across the hybrids Hybrid_1 and Hybrid_2, and hence the indistinguishability holds.
Hybrid_3:

- Ch samples $k \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.

- Ch samples $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$.

- Ch generates the circuit $\tilde{C} \leftarrow \mathsf{iO}(C)$ where C has k hardcoded and on input $r \leftarrow \{0,1\}^{\frac{n}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k, \mathsf{PRG}(r)) \oplus m)$.

- Ch generates ~~$\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{F})$ where $\tilde{F} \leftarrow \mathsf{iO}(\mathsf{PRF.Eval}(k,\cdot))$~~ $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO'.Obf}(1^\lambda, \mathsf{PRF.Eval}(k,\cdot))$ and sends $(\rho_{\mathsf{sk}}, \tilde{C})$ to $\mathcal{A}$.

- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.

- Ch samples $z^{\mathcal{B}}, z^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$ and computes $m^{\mathcal{B}} = \mathsf{PRF.Eval}(k, x^{\mathcal{B}}) \oplus z^{\mathcal{B}}$, $m^{\mathcal{C}} = \mathsf{PRF.Eval}(k, x^{\mathcal{C}}) \oplus z^{\mathcal{C}}$.

- Ch computes $\mathsf{ct}^{\mathcal{B}} = (x^{\mathcal{B}}, z^{\mathcal{B}})$ and $\mathsf{ct}^{\mathcal{C}} = (x^{\mathcal{C}}, z^{\mathcal{C}})$.

- Apply $(\mathcal{B}(\mathsf{ct}^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(y^{\mathcal{B}}, y^{\mathcal{C}})$.

- Output 1 if $y^{\mathcal{B}} = m^{\mathcal{B}}$ and $y^{\mathcal{C}} = m^{\mathcal{C}}$.

Hybrid_3 is just a rewriting of Hybrid_2 in terms of the new unclonable puncturable obfuscation scheme defined as:

- $\mathsf{UPO'.Obf}(1^\lambda, C) = \mathsf{UPO.Obf}(1^\lambda, \tilde{C})$ where $\tilde{C} \leftarrow \mathsf{iO}(C)$, for every circuit C.

- $\mathsf{UPO'.Eval} = \mathsf{UPO.Eval}$.

Note that by Corollary 12, since UPO is a unclonable puncturable obfuscation for any generalized keyed circuit class in $\mathsf{P/poly}$ with respect to $\mathcal{D}_{\mathcal{X}} = \mathcal{U} \times \mathcal{U}$, the product of uniform distribution, so is $\mathsf{UPO'}$.

Next, we give a reduction from Hybrid_3 to an anti-piracy game with uniform and independent challenge distribution (see Figure 18) for $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Puncture}, \mathsf{Verify})$ with respect to $\mathsf{UPO'}$ where Gen on input 1^λ samples a key $k \leftarrow \mathsf{PRF.Gen}(1^\lambda)$ and then constructs the circuit $\tilde{C} \leftarrow \mathsf{iO}(C)$ where C has k hardcoded and on input $r \leftarrow \{0,1\}^{\frac{n}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k, \mathsf{PRG}(r)) \oplus m)$, and finally outputs $(\mathsf{sk}, \mathsf{aux}) = (k, \tilde{C})$. Eval is the same as $\mathsf{PRF.Eval}$; the $\mathsf{Verify}()$ algorithm on input $k, \tilde{C}, x, y$ checks if $\mathsf{PRF.Eval}(k, x) = y$ and if true outputs 1 else 0. Finally, the $\mathsf{Puncture}()$ algorithm on input a key k and a set of input points (x_1, x_2), generates $k_{x_1,x_2} \leftarrow \mathsf{PRF.Puncture}(k, x_1, x_2)$ and outputs $\mathsf{PRF.Eval}(k_{x_1,x_2}, \cdot)$.

Let $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ be an adversary in Hybrid_2 above. Consider the following adversary $(\mathcal{R}_{\mathcal{A}}, \mathcal{R}_{\mathcal{B}}, \mathcal{R}_{\mathcal{C}})$ in $\mathsf{Expt}^{(\mathcal{R}_{\mathcal{A}}, \mathcal{R}_{\mathcal{B}}, \mathcal{R}_{\mathcal{C}}), (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Puncture}, \mathsf{Verify})}(1^\lambda)$ (see Figure 18):

- $\mathcal{R}_{\mathcal{A}}$ on receiving $(\rho_{\mathsf{sk}}, \tilde{C})$ from the challenger Ch in $\mathsf{Expt}^{(\mathcal{R}_{\mathcal{A}}, \mathcal{R}_{\mathcal{B}}, \mathcal{R}_{\mathcal{C}}), (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Puncture}, \mathsf{Verify})}(1^\lambda)$ (see Figure 18), runs $\mathcal{A}$ on it to generate $\sigma_{\mathcal{B},\mathcal{C}}$ and sends the respective registers to $\mathcal{R}_{\mathcal{B}}$ and $\mathcal{R}_{\mathcal{C}}$.

- $\mathcal{R}_{\mathcal{B}}$ (respectively, $\mathcal{R}_{\mathcal{C}}$) on receiving $x^{\mathcal{B}}$ (respectively $x^{\mathcal{C}}$), samples $z^{\mathcal{B}} \xleftarrow{\$} \{0,1\}^n$ (respectively, $z^{\mathcal{C}}$) and runs $\mathcal{B}$ (respectively, $\mathcal{C}$) on $((z^{\mathcal{B}}, x^{\mathcal{B}}), \sigma_{\mathcal{B}})$ (respectively, $((z^{\mathcal{C}}, x^{\mathcal{C}}), \sigma_{\mathcal{C}})$) to get $m^{\mathcal{B}}$ (respectively, $m^{\mathcal{C}}$). $\mathcal{R}_{\mathcal{B}}$ (respectively, $\mathcal{R}_{\mathcal{C}}$) outputs $m^{\mathcal{B}} \oplus z^{\mathcal{B}}$ (respectively, $m^{\mathcal{C}} \oplus z^{\mathcal{C}}$).

Clearly, the event $1 \leftarrow \mathsf{Expt}^{(\mathcal{R}_{\mathcal{A}}, \mathcal{R}_{\mathcal{B}}, \mathcal{R}_{\mathcal{C}}), (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Puncture}, \mathsf{Verify}), \mathsf{UPO'}}(1^\lambda)$ (see Figure 18) exactly corresponds to the event $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ winning the security experiment in Hybrid_3.

By Lemma 65, we know that $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Puncture}, \mathsf{Verify})$ is a puncturable cryptographic scheme. Hence by Lemma 55, for every adversary $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ in Figure 18 against $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Puncture}, \mathsf{Verify})$, there exists a negligible function $\mathsf{negl}()$ such that

$$\Pr\left[1 \leftarrow \mathsf{Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),(\mathsf{Gen},\mathsf{Eval},\mathsf{Puncture},\mathsf{Verify}),\mathsf{UPO}}\left(1^\lambda\right)\right] \leq \mathsf{negl}(\lambda).$$

Hence by the reduction, we conclude that $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ has negligible winning probability in the security experiment in Hybrid_3, which completes the proof.

□

Proposition 66. *The* single decryptor encryption *construction given in Figure 20 satisfies* $\mathcal{D}_{\mathsf{iden\text{-}bit,ind\text{-}msg}}$*-selective* CPA*-style anti-piracy (see Appendix A.2).*

Proof. Let UPO' be a new unclonable puncturable obfuscation scheme defined as:

- $\mathsf{UPO}'.\mathsf{Obf}(1^\lambda, C) = \mathsf{UPO}.\mathsf{Obf}(1^\lambda, \tilde{C})$ where $\tilde{C} \leftarrow \mathsf{iO}(C)$, for every circuit C.
- $\mathsf{UPO}'.\mathsf{Eval} = \mathsf{UPO}.\mathsf{Eval}$.

By Corollary 12, since UPO is a unclonable puncturable obfuscation for any generalized keyed circuit class in $\mathsf{P/poly}$ with respect to the independent challenge distribution $\mathcal{D}_{\mathcal{X}} = \mathcal{U} \times \mathcal{U}$, UPO' also satisfies the same security guarantees.

Let $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ be any adversary in $\mathsf{SelCPA.SDE.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}_{\mathsf{iden\text{-}bit,ind\text{-}msg}}}\left(1^\lambda\right)$ (see Figure 35) against the single decryptor encryption construction in Figure 20. We will do a sequence of hybrids starting from the original anti-piracy experiment $\mathsf{SelCPA.SDE.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}}\left(1^\lambda\right)$ for the single decryptor encryption scheme given in Figure 20, and finally give a reduction to the generalized unclonable puncturable obfuscation security game of UPO' for $\mathcal{F} = \{\mathcal{F}_\lambda\}$, where $\mathcal{F}_\lambda = \{\mathsf{PRF.Eval}(k, \cdot)\}_{k \in \mathsf{Supp}(\mathsf{PRF.Gen}(1^\lambda))}$ with respect to the puncture algorithm $\mathsf{GenPuncture}$ defined as follows: the $\mathsf{GenPuncture}$ algorithm, which takes as input $(k, x_1, x_2, \mu_1, \mu_2)$ and does the following:

- Generates $k_{x_1,x_2} \leftarrow \mathsf{PRF.Puncture}(k, x_1, x_2)$.
- Constructs the circuit $G_{k_{x_1,x_2},x_1,x_2,\mu_1,\mu_2}$ which on input x, outputs $\mathsf{PRF.Eval}(k_{x_1,x_2}, x)$ if $x \notin \{x_1, x_2\}$, and outputs $\mu_1(x_1)$ if $x = x_1$ and $\mu_2(x_2)$ if $x = x_2$.
- Output E.

The changes are marked in blue.

Hybrid_0:

Same as $\mathsf{SelCPA.SDE.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}_{\mathsf{iden\text{-}bit,ind\text{-}msg}}}\left(1^\lambda\right)$ given in Figure 35 for the single decryptor encryption scheme in Figure 20.

- Ch samples $k \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.
- Ch samples $r^{\mathcal{B}}, r^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^{\frac{n}{2}}$ and generates $x^{\mathcal{B}} \leftarrow \mathsf{PRG}(r^{\mathcal{B}})$ and $x^{\mathcal{C}} \leftarrow \mathsf{PRG}(r^{\mathcal{C}})$ as well as generates $y^{\mathcal{B}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{B}})$ and $y^{\mathcal{C}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{C}})$.
- Ch generates the circuit $\tilde{C} \leftarrow \mathsf{iO}(C)$ where C has k hardcoded and on input $r \leftarrow \{0,1\}^{\frac{n}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k, \mathsf{PRG}(r)) \oplus m)$.

- Ch generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{F})$ where $\tilde{F} \leftarrow \mathsf{iO}(\mathsf{PRF.Eval}(k, \cdot))$ and sends $(\rho_{\mathsf{sk}}, \tilde{C})$ to $\mathcal{A}$.
- $\mathcal{A}$ sends two messages $(m_0^{\mathcal{B}}, m_1^{\mathcal{B}}, m_0^{\mathcal{C}}, m_1^{\mathcal{C}})$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- Ch samples $b \xleftarrow{\$} \{0,1\}$.
- Ch computes $\mathsf{ct}^{\mathcal{B}} = (x^{\mathcal{B}}, z^{\mathcal{B}})$ and $\mathsf{ct}^{\mathcal{C}} = (x^{\mathcal{C}}, z^{\mathcal{C}})$ where $z^{\mathcal{B}} = y^{\mathcal{B}} \oplus m_b^{\mathcal{B}}$ and $z^{\mathcal{C}} = y^{\mathcal{C}} \oplus m_b^{\mathcal{C}}$.
- Apply $(\mathcal{B}(\mathsf{ct}^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.
- Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$.

Hybrid_1:
This is the same as Hybrid_0 up to re-ordering some of the steps performed by the Ch without affecting view of the adversary.

- Ch samples $k \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.
- Ch samples $b \xleftarrow{\$} \{0,1\}$.
- Ch samples $r^{\mathcal{B}}, r^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^{\frac{n}{2}}$ and generates $x^{\mathcal{B}} \leftarrow \mathsf{PRG}(r^{\mathcal{B}})$ and $x^{\mathcal{C}} \leftarrow \mathsf{PRG}(r^{\mathcal{C}})$ as well as generates $y^{\mathcal{B}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{B}})$ and $y^{\mathcal{C}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{C}})$.
- Ch generates the circuit $\tilde{C} \leftarrow \mathsf{iO}(C)$ where C has k hardcoded and on input $r \leftarrow \{0,1\}^{\frac{n}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k, \mathsf{PRG}(r)) \oplus m)$.
- Ch generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{F})$ where $\tilde{F} \leftarrow \mathsf{iO}(\mathsf{PRF.Eval}(k, \cdot))$ and sends $(\rho_{\mathsf{sk}}, \tilde{C})$ to $\mathcal{A}$.
- $\mathcal{A}$ sends two messages $(m_0^{\mathcal{B}}, m_1^{\mathcal{B}}, m_0^{\mathcal{C}}, m_1^{\mathcal{C}})$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- ~~Ch samples $b \xleftarrow{\$} \{0,1\}$.~~
- Ch computes $\mathsf{ct}^{\mathcal{B}} = (x^{\mathcal{B}}, z^{\mathcal{B}})$ and $\mathsf{ct}^{\mathcal{C}} = (x^{\mathcal{C}}, z^{\mathcal{C}})$ where $z^{\mathcal{B}} = y^{\mathcal{B}} \oplus m_b^{\mathcal{B}}$ and $z^{\mathcal{C}} = y^{\mathcal{C}} \oplus m_b^{\mathcal{C}}$.
- Apply $(\mathcal{B}(\mathsf{ct}^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.
- Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$.

Hybrid_2:

- Ch samples $k \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.
- Ch samples $b \xleftarrow{\$} \{0,1\}$.
- Ch samples ~~$r^{\mathcal{B}}, r^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^{\frac{n}{2}}$ and generates $x^{\mathcal{B}} \leftarrow \mathsf{PRG}(r^{\mathcal{B}})$ and $x^{\mathcal{C}} \leftarrow \mathsf{PRG}(r^{\mathcal{C}})$~~ $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$ as well as generates $y^{\mathcal{B}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{B}})$ and $y^{\mathcal{C}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{C}})$.

- Ch generates the circuit $\tilde{C} \leftarrow \mathsf{iO}(C)$ where C has k hardcoded and on input $r \leftarrow \{0,1\}^{\frac{n}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k, \mathsf{PRG}(r)) \oplus m)$.
- Ch generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{F})$ where $\tilde{F} \leftarrow \mathsf{iO}(\mathsf{PRF.Eval}(k, \cdot))$ and sends $(\rho_{\mathsf{sk}}, \tilde{C})$ to $\mathcal{A}$.
- $\mathcal{A}$ sends two messages $(m_0^{\mathcal{B}}, m_1^{\mathcal{B}}, m_0^{\mathcal{C}}, m_1^{\mathcal{C}})$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- Ch samples $b \xleftarrow{\$} \{0,1\}$.
- Ch computes $\mathsf{ct}^{\mathcal{B}} = (x^{\mathcal{B}}, z^{\mathcal{B}})$ and $\mathsf{ct}^{\mathcal{C}} = (x^{\mathcal{C}}, z^{\mathcal{C}})$ where $z^{\mathcal{B}} = y^{\mathcal{B}} \oplus m_b^{\mathcal{B}}$ and $z^{\mathcal{C}} = y^{\mathcal{C}} \oplus m_b^{\mathcal{C}}$.
- Apply $(\mathcal{B}(\mathsf{ct}^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.
- Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$.

The indistinguishability between Hybrid_1 and Hybrid_2 follows from the pseudorandomness of PRG.
Hybrid_3:

- Ch samples $k \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.
- Ch samples $b \xleftarrow{\$} \{0,1\}$.
- Ch samples $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$ as well as generates $y^{\mathcal{B}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{B}})$ and $y^{\mathcal{C}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{C}})$.
- Ch generates $k_{x^{\mathcal{B}},x^{\mathcal{C}}} \leftarrow \mathsf{PRF.Puncture}(k, \{x^{\mathcal{B}}, x^{\mathcal{C}}\})$.
- Ch generates the circuit $\tilde{C} \leftarrow \mathsf{iO}(C)$ where C has k hardcoded and on input $r \leftarrow \{0,1\}^{\frac{n}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k, \mathsf{PRG}(r)) \oplus m)$. where C is constructed depending on the bit b as follows. If $b = 0$ (respectively, $b = 1$), C has k (respectively, $k_{x^{\mathcal{B}},x^{\mathcal{C}}}$) hardcoded and on input $r \leftarrow \{0,1\}^{\frac{n}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k, \mathsf{PRG}(r)) \oplus m)$ (respectively, $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k_{x^{\mathcal{B}},x^{\mathcal{C}}}, \mathsf{PRG}(r)) \oplus m)$).
- Ch generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{F})$ where $\tilde{F} \leftarrow \mathsf{iO}(\mathsf{PRF.Eval}(k, \cdot))$ and sends $(\rho_{\mathsf{sk}}, \tilde{C})$ to $\mathcal{A}$.
- $\mathcal{A}$ sends two messages $(m_0^{\mathcal{B}}, m_1^{\mathcal{B}}, m_0^{\mathcal{C}}, m_1^{\mathcal{C}})$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- Ch computes $\mathsf{ct}^{\mathcal{B}} = (x^{\mathcal{B}}, z^{\mathcal{B}})$ and $\mathsf{ct}^{\mathcal{C}} = (x^{\mathcal{C}}, z^{\mathcal{C}})$ where $z^{\mathcal{B}} = y^{\mathcal{B}} \oplus m_b^{\mathcal{B}}$ and $z^{\mathcal{C}} = y^{\mathcal{C}} \oplus m_b^{\mathcal{C}}$.
- Apply $(\mathcal{B}(\mathsf{ct}^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.
- Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$.

The proof of indistinguishability between Hybrid_2 and Hybrid_3 is as follows. Note that $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$ and $\mathsf{Supp}(\mathsf{PRG}) \subset \{0,1\}^n$ has size $2^{\frac{n}{2}}$, and hence is a negligible fraction of $\{0,1\}^n$. Hence, with overwhelming probability $x^{\mathcal{B}}, x^{\mathcal{C}} \notin \mathsf{Supp}(\mathsf{PRG})$. Therefore with overwhelming probability, C as in Hybrid_0 never computes $\mathsf{PRF.Eval}(k, \cdot)$ on $x^{\mathcal{B}}$ or $x^{\mathcal{C}}$ on any input query. Hence, replacing k with k_{x_1,x_2} inside C in the $b = 1$ case of the security experiment does not change the functionality of C, by the puncturing correctness of PRF. Therefore, indistinguishability holds by the iO guarantee.
Hybrid_4:

- Ch samples $k \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.
- Ch samples $b \xleftarrow{\$} \{0,1\}$.
- Ch samples $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$ as well as generates $y^{\mathcal{B}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{B}})$ and $y^{\mathcal{C}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{C}})$.
- Ch generates $k_{x^{\mathcal{B}},x^{\mathcal{C}}} \leftarrow \mathsf{PRF.Puncture}(k, \{x^{\mathcal{B}}, x^{\mathcal{C}}\})$.
- Ch generates the circuit $\tilde{C} \leftarrow \mathsf{iO}(C)$ where C is constructed depending on the bit b as follows. If $b = 0$ (respectively, $b = 1$), C has k (respectively, $k_{x^{\mathcal{B}},x^{\mathcal{C}}}$) hardcoded and on input $r \leftarrow \{0,1\}^{\frac{n}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k, \mathsf{PRG}(r)) \oplus m)$ (respectively, $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k_{x^{\mathcal{B}},x^{\mathcal{C}}}, \mathsf{PRG}(r)) \oplus m)$).
- If $b = 0$, Ch generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{F})$ where $\tilde{F} \leftarrow \mathsf{iO}(\mathsf{PRF.Eval}(k, \cdot))$, else, if $b = 1$, generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{W})$, where $\tilde{W} \leftarrow \mathsf{iO}(W)$ and W is as depicted in Figure 22 and sends $(\rho_{\mathsf{sk}}, \tilde{C})$ to $\mathcal{A}$.
- $\mathcal{A}$ sends two messages $(m_0^{\mathcal{B}}, m_1^{\mathcal{B}}, m_0^{\mathcal{C}}, m_1^{\mathcal{C}})$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- Ch computes $\mathsf{ct}^{\mathcal{B}} = (x^{\mathcal{B}}, z^{\mathcal{B}})$ and $\mathsf{ct}^{\mathcal{C}} = (x^{\mathcal{C}}, z^{\mathcal{C}})$ where $z^{\mathcal{B}} = y^{\mathcal{B}} \oplus m_b^{\mathcal{B}}$ and $z^{\mathcal{C}} = y^{\mathcal{C}} \oplus m_b^{\mathcal{C}}$.
- Apply $(\mathcal{B}(\mathsf{ct}^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.
- Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$.

Clearly, W and $\mathsf{PRF.Eval}(k, \cdot)$ has the same functionality and therefore indistinguishability holds by iO guarantees.
Hybrid_5:

- Ch samples $k \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.
- Ch samples $b \xleftarrow{\$} \{0,1\}$.
- Ch samples $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$ as well as generates $y^{\mathcal{B}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{B}})$, $y^{\mathcal{C}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{C}})$ if $b = 0$; and $y^{\mathcal{B}} \xleftarrow{\$} \{0,1\}^n$, and $y^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$ if $b = 1$.
- Ch generates $k_{x^{\mathcal{B}},x^{\mathcal{C}}} \leftarrow \mathsf{PRF.Puncture}(k, \{x^{\mathcal{B}}, x^{\mathcal{C}}\})$.

$\underline{W}$:

Hardcoded keys $k_{x^{\mathcal{B}},x^{\mathcal{C}}}, y^{\mathcal{B}}, y^{\mathcal{C}}$. On input: x.

- If $x = x^{\mathcal{B}}$, output $y^{\mathcal{B}}$.
- Else if, $x = x^{\mathcal{C}}$, output $y^{\mathcal{C}}$.
- Else, run $\mathsf{PRF.Eval}(k_{x^{\mathcal{B}},x^{\mathcal{C}}}, x)$ and output the result.

Figure 22: Circuit W in Hybrid_4

- Ch generates the circuit $\tilde{C} \leftarrow \mathsf{iO}(C)$ where C is constructed depending on the bit b as follows. If $b = 0$ (respectively, $b = 1$), C has k (respectively, $k_{x^{\mathcal{B}},x^{\mathcal{C}}}$) hardcoded and on input $r \leftarrow \{0,1\}^{\frac{n}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k, \mathsf{PRG}(r)) \oplus m)$ (respectively, $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k_{x^{\mathcal{B}},x^{\mathcal{C}}}, \mathsf{PRG}(r)) \oplus m)$).
- If $b = 0$, Ch generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{F})$ where $\tilde{F} \leftarrow \mathsf{iO}(\mathsf{PRF.Eval}(k, \cdot))$, else, if $b = 1$, generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{W})$, where $\tilde{W} \leftarrow \mathsf{iO}(W)$ and W is as depicted in Figure 22 and sends $(\rho_{\mathsf{sk}}, \mathsf{iO}(C))$ to $\mathcal{A}$.
- $\mathcal{A}$ sends two messages $(m_0^{\mathcal{B}}, m_1^{\mathcal{B}}, m_0^{\mathcal{C}}, m_1^{\mathcal{C}})$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- Ch computes $\mathsf{ct}^{\mathcal{B}} = (x^{\mathcal{B}}, z^{\mathcal{B}})$ and $\mathsf{ct}^{\mathcal{C}} = (x^{\mathcal{C}}, z^{\mathcal{C}})$ where $z^{\mathcal{B}} = y^{\mathcal{B}} \oplus m_b^{\mathcal{B}}$ and $z^{\mathcal{C}} = y^{\mathcal{C}} \oplus m_b^{\mathcal{C}}$.
- Apply $(\mathcal{B}(\mathsf{ct}^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.
- Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$.

Since the views of the adversary $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ in $b = 1$ case in hybrids Hybrid_4 and Hybrid_5 are only dependent on $k_{x^{\mathcal{B}},x^{\mathcal{C}}}$, the indistinguishability between Hybrid_4 and Hybrid_5 holds by the puncturing security of PRF.
Hybrid_6:

- Ch samples $k \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.
- Ch samples $b \xleftarrow{\$} \{0,1\}$.
- Ch samples $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$ ~~as well as generates $y^{\mathcal{B}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{B}})$, $y^{\mathcal{C}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{C}})$ if $b = 0$; and $y^{\mathcal{B}} \xleftarrow{\$} \{0,1\}^n$, and $y^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$ if $b = 1$~~.
- Ch generates $k_{x^{\mathcal{B}},x^{\mathcal{C}}} \leftarrow \mathsf{PRF.Puncture}(k, \{x^{\mathcal{B}}, x^{\mathcal{C}}\})$.

- Ch generates the circuit $\tilde{C} \leftarrow \mathsf{iO}(C)$ where C is constructed depending on the bit b as follows. If $b = 0$ (respectively, $b = 1$), C has k (respectively, $k_{x^{\mathcal{B}},x^{\mathcal{C}}}$) hardcoded and on input $r \leftarrow \{0,1\}^{\frac{n}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k, \mathsf{PRG}(r)) \oplus m)$ (respectively, $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k_{x^{\mathcal{B}},x^{\mathcal{C}}}, \mathsf{PRG}(r)) \oplus m)$).

- If $b = 0$, Ch generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{F})$ where $\tilde{F} \leftarrow \mathsf{iO}(\mathsf{PRF.Eval}(k, \cdot))$, else, if $b = 1$, generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{W})$, where $\tilde{W} \leftarrow \mathsf{iO}(W)$ and W is as depicted in Figure 22 and sends $(\rho_{\mathsf{sk}}, \mathsf{iO}(C))$ to $\mathcal{A}$.

- $\mathcal{A}$ sends two messages $(m_0^{\mathcal{B}}, m_1^{\mathcal{B}}, m_0^{\mathcal{C}}, m_1^{\mathcal{C}})$.

- If $b = 1$, Ch samples $u^{\mathcal{B}}, u^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$ and computes $y^{\mathcal{B}} = u^{\mathcal{B}} \oplus m_0^{\mathcal{B}} \oplus m_1^{\mathcal{B}}$ and $y^{\mathcal{C}} = u^{\mathcal{C}} \oplus m_0^{\mathcal{C}} \oplus m_1^{\mathcal{C}}$, else if $b = 0$, Ch generates $y^{\mathcal{B}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{B}})$, $y^{\mathcal{C}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{C}})$.

- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.

- Ch computes $\mathsf{ct}^{\mathcal{B}} = (x^{\mathcal{B}}, z^{\mathcal{B}})$ and $\mathsf{ct}^{\mathcal{C}} = (x^{\mathcal{C}}, z^{\mathcal{C}})$ where ~~$z^{\mathcal{B}} = y^{\mathcal{B}} \oplus m_b^{\mathcal{B}}$ and $z^{\mathcal{C}} = y^{\mathcal{C}} \oplus m_b^{\mathcal{C}}$~~ $z^{\mathcal{B}} = y^{\mathcal{B}} \oplus m_0^{\mathcal{B}}$ and $z^{\mathcal{C}} = y^{\mathcal{C}} \oplus m_0^{\mathcal{C}}$ if $b = 0$, and $z^{\mathcal{B}} = y^{\mathcal{B}} \oplus m_1^{\mathcal{B}}$ and $z^{\mathcal{C}} = y^{\mathcal{C}} \oplus m_1^{\mathcal{C}}$ if $b = 1$.

- Apply $(\mathcal{B}(\mathsf{ct}^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.

- Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$.

The indistinguishability between Hybrid_5 and Hybrid_6 since we did not change the distribution on $y^{\mathcal{B}}$, $y^{\mathcal{C}}$ in both the cases $b = 0$ and $b = 1$, and hence we did not change the distribution on $z^{\mathcal{B}}$, $z^{\mathcal{C}}$ in both the $b = 0$ and the $b = 1$ cases across the hybrids Hybrid_5 and Hybrid_6.
Hybrid_7:

- Ch samples $k \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.

- Ch samples $b \xleftarrow{\$} \{0,1\}$.

- Ch samples $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$.

- Ch generates $k_{x^{\mathcal{B}},x^{\mathcal{C}}} \leftarrow \mathsf{PRF.Puncture}(k, \{x^{\mathcal{B}}, x^{\mathcal{C}}\})$.

- Ch generates the circuit $\tilde{C} \leftarrow \mathsf{iO}(C)$ where C is constructed depending on the bit b as follows. If $b = 0$ (respectively, $b = 1$), C has k (respectively, $k_{x^{\mathcal{B}},x^{\mathcal{C}}}$) hardcoded and on input $r \leftarrow \{0,1\}^{\frac{n}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k, \mathsf{PRG}(r)) \oplus m)$ (respectively, $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k_{x^{\mathcal{B}},x^{\mathcal{C}}}, \mathsf{PRG}(r)) \oplus m)$).

- If $b = 0$, Ch generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{F})$ where $\tilde{F} \leftarrow \mathsf{iO}(\mathsf{PRF.Eval}(k, \cdot))$, else, if $b = 1$, generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{W})$, where $\tilde{W} \leftarrow \mathsf{iO}(W)$ and W is as depicted in Figure 23 and sends $(\rho_{\mathsf{sk}}, \mathsf{iO}(C))$ to $\mathcal{A}$.

- $\mathcal{A}$ sends two messages $(m_0^{\mathcal{B}}, m_1^{\mathcal{B}}, m_0^{\mathcal{C}}, m_1^{\mathcal{C}})$.

- If $b = 1$, Ch samples $u^{\mathcal{B}}, u^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$ ~~and computes $y^{\mathcal{B}} = u^{\mathcal{B}} \oplus m_0^{\mathcal{B}} \oplus m_1^{\mathcal{B}}$ and $y^{\mathcal{C}} = u^{\mathcal{C}} \oplus m_0^{\mathcal{C}} \oplus m_1^{\mathcal{C}}$~~, else if $b = 0$, Ch generates $y^{\mathcal{B}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{B}})$, $y^{\mathcal{C}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{C}})$.

- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.

- Ch computes $\mathsf{ct}^{\mathcal{B}} = (x^{\mathcal{B}}, z^{\mathcal{B}})$ and $\mathsf{ct}^{\mathcal{C}} = (x^{\mathcal{C}}, z^{\mathcal{C}})$ where $z^{\mathcal{B}} = y^{\mathcal{B}} \oplus m_0^{\mathcal{B}}$ and $z^{\mathcal{C}} = y^{\mathcal{C}} \oplus m_0^{\mathcal{C}}$ if $b = 0$, and ~~$z^{\mathcal{B}} = y^{\mathcal{B}} \oplus m_1^{\mathcal{B}}$ and $z^{\mathcal{C}} = y^{\mathcal{C}} \oplus m_1^{\mathcal{C}}$~~ $z^{\mathcal{B}} = u^{\mathcal{B}} \oplus m_0^{\mathcal{B}}$ and $z^{\mathcal{C}} = u^{\mathcal{C}} \oplus m_0^{\mathcal{C}}$ if $b = 1$.

- Apply $(\mathcal{B}(\mathsf{ct}^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.

- Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$.

<u>W</u>:

Hardcoded keys $k_{x^{\mathcal{B}},x^{\mathcal{C}}}$ ~~$y^{\mathcal{B}}, y^{\mathcal{C}}$~~, $u^{\mathcal{B}} \oplus m_0^{\mathcal{B}} \oplus m_1^{\mathcal{B}}, u^{\mathcal{C}} \oplus m_0^{\mathcal{C}} \oplus m_1^{\mathcal{C}}$. On input: x.

- If $x = x^{\mathcal{B}}$, output ~~$y^{\mathcal{B}}$~~ $u^{\mathcal{B}} \oplus m_0^{\mathcal{B}} \oplus m_1^{\mathcal{B}}$.
- Else if, $x = x^{\mathcal{C}}$, output ~~$y^{\mathcal{C}}$~~ $u^{\mathcal{C}} \oplus m_0^{\mathcal{C}} \oplus m_1^{\mathcal{C}}$.
- Else, run $\mathsf{PRF.Eval}(k_{x^{\mathcal{B}},x^{\mathcal{C}}}, x)$ and output the result.

Figure 23: Circuit W in Hybrid_7

The indistinguishability between Hybrid_6 and Hybrid_7 holds because, in Hybrid_7, we just rewrote $y^{\mathcal{B}}$ and $y^{\mathcal{C}}$ wherever it appeared in the $b = 1$ case of Hybrid_6 in terms of $u^{\mathcal{B}}$ and $u^{\mathcal{C}}$, respectively.
Hybrid_8:

- Ch samples $k \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.

- Ch samples $b \xleftarrow{\$} \{0,1\}$.

- Ch samples $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$.

- Ch generates $k_{x^{\mathcal{B}},x^{\mathcal{C}}} \leftarrow \mathsf{PRF.Puncture}(k, \{x^{\mathcal{B}}, x^{\mathcal{C}}\})$.

- Ch generates the circuit $\tilde{C} \leftarrow \mathsf{iO}(C)$ where C is constructed depending on the bit b as follows. If $b = 0$ (respectively, $b = 1$), C has k (respectively, $k_{x^{\mathcal{B}},x^{\mathcal{C}}}$) hardcoded and on input $r \leftarrow \{0,1\}^{\frac{n}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k, \mathsf{PRG}(r)) \oplus m)$ (respectively, $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k_{x^{\mathcal{B}},x^{\mathcal{C}}}, \mathsf{PRG}(r)) \oplus m)$).

- If $b = 0$, Ch generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{F})$ where $\tilde{F} \leftarrow \mathsf{iO}(\mathsf{PRF.Eval}(k, \cdot))$, else, if $b = 1$, generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{W})$, where $\tilde{W} \leftarrow \mathsf{iO}(W)$ and W is as depicted in Figure 23 and sends $(\rho_{\mathsf{sk}}, \mathsf{iO}(C))$ to $\mathcal{A}$.

- $\mathcal{A}$ sends two messages $(m_0^{\mathcal{B}}, m_1^{\mathcal{B}}, m_0^{\mathcal{C}}, m_1^{\mathcal{C}})$.
- If $b = 1$, Ch ~~samples $u^{\mathcal{B}}, u^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$~~ generates $u^{\mathcal{B}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{B}})$, $u^{\mathcal{C}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{C}})$, else if $b = 0$, Ch generates $y^{\mathcal{B}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{B}})$, $y^{\mathcal{C}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{C}})$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- Ch computes $\mathsf{ct}^{\mathcal{B}} = (x^{\mathcal{B}}, z^{\mathcal{B}})$ and $\mathsf{ct}^{\mathcal{C}} = (x^{\mathcal{C}}, z^{\mathcal{C}})$ where $z^{\mathcal{B}} = y^{\mathcal{B}} \oplus m_0^{\mathcal{B}}$ and $z^{\mathcal{C}} = y^{\mathcal{C}} \oplus m_0^{\mathcal{C}}$ if $b = 0$, and $z^{\mathcal{B}} = u^{\mathcal{B}} \oplus m_0^{\mathcal{B}}$ and $z^{\mathcal{C}} = u^{\mathcal{C}} \oplus m_0^{\mathcal{C}}$ if $b = 1$.
- Apply $(\mathcal{B}(\mathsf{ct}^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.
- Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$.

Since the views of the adversary $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ in $b = 1$ case in hybrids Hybrid_7 and Hybrid_8 are only dependent on $k_{x^{\mathcal{B}},x^{\mathcal{C}}}$, the indistinguishability between Hybrid_7 and Hybrid_8 holds by the puncturing security of PRF.
Hybrid_9:

- Ch samples $k \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.
- Ch samples $b \xleftarrow{\$} \{0,1\}$.
- Ch samples $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$.
- Ch generates $k_{x^{\mathcal{B}},x^{\mathcal{C}}} \leftarrow \mathsf{PRF.Puncture}(k, \{x^{\mathcal{B}}, x^{\mathcal{C}}\})$.
- Ch generates the circuit $\tilde{C} \leftarrow \mathsf{iO}(C)$ where C is constructed depending on the bit b as follows. If $b = 0$ (respectively, $b = 1$), C has k (respectively, $k_{x^{\mathcal{B}},x^{\mathcal{C}}}$) hardcoded and on input $r \leftarrow \{0,1\}^{\frac{n}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k, \mathsf{PRG}(r)) \oplus m)$ (respectively, $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k_{x^{\mathcal{B}},x^{\mathcal{C}}}, \mathsf{PRG}(r)) \oplus m)$).
- If $b = 0$, Ch generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{F})$ where $\tilde{F} \leftarrow \mathsf{iO}(\mathsf{PRF.Eval}(k, \cdot))$, else, if $b = 1$, generates the circuits $\mu_{k, m_0^{\mathcal{B}} \oplus m_1^{\mathcal{B}}}$ and $\mu_{k, m_0^{\mathcal{C}} \oplus m_1^{\mathcal{C}}}$ which on any input x output $\mathsf{PRF.Eval}(k, x) \oplus m_0^{\mathcal{B}} \oplus m_1^{\mathcal{B}}$ and $\mathsf{PRF.Eval}(k, x) \oplus m_0^{\mathcal{C}} \oplus m_1^{\mathcal{C}}$ respectively, and also generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{W})$, where $\tilde{W} \leftarrow \mathsf{iO}(W)$ and W is as depicted in Figure 24 and sends $(\rho_{\mathsf{sk}}, \tilde{C})$ to $\mathcal{A}$.
- $\mathcal{A}$ sends two messages $(m_0^{\mathcal{B}}, m_1^{\mathcal{B}}, m_0^{\mathcal{C}}, m_1^{\mathcal{C}})$.
- If $b = 1$, Ch generates $u^{\mathcal{B}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{B}})$, $u^{\mathcal{C}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{C}})$, else if $b = 0$, Ch generates $y^{\mathcal{B}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{B}})$, $y^{\mathcal{C}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{C}})$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- Ch computes $\mathsf{ct}^{\mathcal{B}} = (x^{\mathcal{B}}, z^{\mathcal{B}})$ and $\mathsf{ct}^{\mathcal{C}} = (x^{\mathcal{C}}, z^{\mathcal{C}})$ where $z^{\mathcal{B}} = y^{\mathcal{B}} \oplus m_0^{\mathcal{B}}$ and $z^{\mathcal{C}} = y^{\mathcal{C}} \oplus m_0^{\mathcal{C}}$ if $b = 0$, and $z^{\mathcal{B}} = u^{\mathcal{B}} \oplus m_0^{\mathcal{B}}$ and $z^{\mathcal{C}} = u^{\mathcal{C}} \oplus m_0^{\mathcal{C}}$ if $b = 1$.
- Apply $(\mathcal{B}(\mathsf{ct}^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.
- Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$.

$\underline{W}$:

Hardcoded keys $k_{x^{\mathcal{B}},x^{\mathcal{C}}}, u^{\mathcal{B}} \oplus m_0^{\mathcal{B}} \oplus m_1^{\mathcal{B}}, u^{\mathcal{C}} \oplus m_0^{\mathcal{C}} \oplus m_1^{\mathcal{C}}, \mu_{k,m_0^{\mathcal{B}} \oplus m_1^{\mathcal{B}}}, \mu_{k,m_0^{\mathcal{C}} \oplus m_1^{\mathcal{C}}}$. On input: x.

- If $x = x^{\mathcal{B}}$, output $u^{\mathcal{B}} \oplus m_0^{\mathcal{B}} \oplus m_1^{\mathcal{B}}$ $\mu_{k,m_0^{\mathcal{B}} \oplus m_1^{\mathcal{B}}}(x)$.
- Else if, $x = x^{\mathcal{C}}$, output $u^{\mathcal{C}} \oplus m_0^{\mathcal{C}} \oplus m_1^{\mathcal{C}}$ $\mu_{k,m_0^{\mathcal{C}} \oplus m_1^{\mathcal{C}}}(x)$.
- Else, run $\mathsf{PRF.Eval}(k_{x^{\mathcal{B}},x^{\mathcal{C}}}, x)$ and output the result.

Figure 24: Circuit W in Hybrid_9

The functionality of W did not change due to the changes made across hybrids Hybrid_8 and Hybrid_9, and hence by iO guarantees, the indistinguishability between Hybrid_8 and Hybrid_9 holds.
Hybrid_{10}:

- Ch samples $k \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.
- Ch samples $b \xleftarrow{\$} \{0,1\}$.
- Ch samples $x^{\mathcal{B}}, x^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$.
- Ch generates $k_{x^{\mathcal{B}},x^{\mathcal{C}}} \leftarrow \mathsf{PRF.Puncture}(k, \{x^{\mathcal{B}}, x^{\mathcal{C}}\})$.
- Ch generates the circuit $\tilde{C} \leftarrow \mathsf{iO}(C)$ where C is constructed depending on the bit b as follows. If $b = 0$ (respectively, $b = 1$), C has k (respectively, $k_{x^{\mathcal{B}},x^{\mathcal{C}}}$) hardcoded and on input $r \leftarrow \{0,1\}^{\frac{n}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k, \mathsf{PRG}(r)) \oplus m)$ (respectively, $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k_{x^{\mathcal{B}},x^{\mathcal{C}}}, \mathsf{PRG}(r)) \oplus m)$). where C has k hardcoded and on input $r \leftarrow \{0,1\}^{\frac{n}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k, \mathsf{PRG}(r)) \oplus m)$.
- If $b = 0$, Ch generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{F})$ where $\tilde{F} \leftarrow \mathsf{iO}(\mathsf{PRF.Eval}(k, \cdot))$, else, if $b = 1$, generates the circuits $\mu_{k,m_0^{\mathcal{B}} \oplus m_1^{\mathcal{B}}}$ and $\mu_{k,m_0^{\mathcal{C}} \oplus m_1^{\mathcal{C}}}$ which on any input x output $\mathsf{PRF.Eval}(k, x) \oplus m_0^{\mathcal{B}} \oplus m_1^{\mathcal{B}}$ and $\mathsf{PRF.Eval}(k, x) \oplus m_0^{\mathcal{C}} \oplus m_1^{\mathcal{C}}$ respectively, and also generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{UPO.Obf}(1^\lambda, \tilde{W})$, where $\tilde{W} \leftarrow \mathsf{iO}(W)$ and W is as depicted in Figure 24 and sends $(\rho_{\mathsf{sk}}, \tilde{C})$ to $\mathcal{A}$.
- $\mathcal{A}$ sends two messages $(m_0^{\mathcal{B}}, m_1^{\mathcal{B}}, m_0^{\mathcal{C}}, m_1^{\mathcal{C}})$.
- If $b = 1$, Ch generates $u^{\mathcal{B}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{B}}), u^{\mathcal{C}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{C}})$, else if $b = 0$, Ch generates $y^{\mathcal{B}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{B}})$, $y^{\mathcal{C}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{C}})$. Ch generates $u^{\mathcal{B}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{B}}), u^{\mathcal{C}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{C}})$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.

- ~~Ch computes $\mathsf{ct}^{\mathcal{B}} = (x^{\mathcal{B}}, z^{\mathcal{B}})$ and $\mathsf{ct}^{\mathcal{C}} = (x^{\mathcal{C}}, z^{\mathcal{C}})$ where $z^{\mathcal{B}} = y^{\mathcal{B}} \oplus m_0^{\mathcal{B}}$ and $z^{\mathcal{C}} = y^{\mathcal{C}} \oplus m_0^{\mathcal{C}}$ if $b = 0$, and $z^{\mathcal{B}} = u^{\mathcal{B}} \oplus m_0^{\mathcal{B}}$ and $z^{\mathcal{C}} = u^{\mathcal{C}} \oplus m_0^{\mathcal{C}}$ if $b = 1$.~~ Ch computes $\mathsf{ct}^{\mathcal{B}} = (x^{\mathcal{B}}, z^{\mathcal{B}})$ and $\mathsf{ct}^{\mathcal{C}} = (x^{\mathcal{C}}, z^{\mathcal{C}})$ where $z^{\mathcal{B}} = u^{\mathcal{B}} \oplus m_0^{\mathcal{B}}$ and $z^{\mathcal{C}} = u^{\mathcal{C}} \oplus m_0^{\mathcal{C}}$.

- Apply $(\mathcal{B}(\mathsf{ct}^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.

- Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$.

Note that $y^{\mathcal{B}}$ and $y^{\mathcal{C}}$ are defined only in the $b = 0$ case, and $u^{\mathcal{B}}$ and $u^{\mathcal{C}}$ are defined only in the $b = 1$ case in Hybrid_9. However, replacing $y^{\mathcal{B}}$, $y^{\mathcal{C}}$ in the $b = 0$ by $u^{\mathcal{B}}$, $u^{\mathcal{C}}$ (as defined in $b = 1$ case) does not change the global distribution of the experiment in $b = 0$ case. Therefore, replacing $y^{\mathcal{B}}, y^{\mathcal{C}}$ in $b = 0$ with $u^{\mathcal{B}}, u^{\mathcal{C}}$ (as defined in the $b = 1$ case) in Hybrid_9, does not change the security experiment and hence, Hybrid_9 and Hybrid_{10} have the same success probability.

Finally, we give a reduction from Hybrid_{10} to the generalized unclonable puncturable obfuscation security experiment (see fig. 3) of UPO' for $\mathfrak{C} = \{\mathfrak{C}_\lambda\}$, where $\mathfrak{C}_\lambda = \{\mathsf{PRF.Eval}(k, \cdot)\}_{k \in \mathsf{Supp}(\mathsf{PRF.Gen}(1^\lambda))}$ with respect to the puncture algorithm $\mathsf{GenPuncture}$ defined at the begining of the proof.

Let $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ be an adversary in Hybrid_{10} above. Consider the following non-local adversary $(\mathcal{R}_{\mathcal{A}}, \mathcal{R}_{\mathcal{B}}, \mathcal{R}_{\mathcal{C}})$:

- $\mathcal{R}_{\mathcal{A}}$ gets a pair of messages $m_0^{\mathcal{B}}, m_1^{\mathcal{B}}, m_0^{\mathcal{C}}, m_1^{\mathcal{C}} \leftarrow \mathcal{A}(1^\lambda)$ and samples a key $k \leftarrow \mathsf{PRF.Gen}(1^\lambda)$ and constructs the circuits $\mu_{k, m_0^{\mathcal{B}} \oplus m_1^{\mathcal{B}}}$ and $\mu_{k, m_0^{\mathcal{C}} \oplus m_1^{\mathcal{C}}}$ which on any input x outputs $\mathsf{PRF.Eval}(k, x) \oplus m_0^{\mathcal{B}} \oplus m_1^{\mathcal{B}}$ and $\mathsf{PRF.Eval}(k, x) \oplus m_0^{\mathcal{C}} \oplus m_1^{\mathcal{C}}$ respectively, and sends $k, \mu_{\mathcal{B}}, \mu_{\mathcal{C}}$ to Ch where $\mu_{\mathcal{B}} = \mu_{k, m_0^{\mathcal{B}} \oplus m_1^{\mathcal{B}}}$ and $\mu_{\mathcal{C}} = \mu_{k, m_0^{\mathcal{C}} \oplus m_1^{\mathcal{C}}}$.

- $\mathcal{R}_{\mathcal{A}}$ also constructs the circuit $\tilde{C} \leftarrow \mathsf{iO}(C)$ where C has k hardcoded and on input $r \leftarrow \{0,1\}^{\frac{n}{2}}$ (the input space of PRG) and a message $m \in \{0,1\}^n$, outputs $(\mathsf{PRG}(r), \mathsf{PRF.Eval}(k, \mathsf{PRG}(r)) \oplus m)$.

- On getting ρ from Ch, $\mathcal{R}_{\mathcal{A}}$ feeds $\rho, \tilde{C}$ to $\mathcal{A}$ and gets back a state $\sigma_{\mathcal{B},\mathcal{C}}$. $\mathcal{R}_{\mathcal{A}}$ then sends the respective registers of $\sigma_{\mathcal{B},\mathcal{C}}$ to $\mathcal{R}_{\mathcal{B}}$ and $\mathcal{R}_{\mathcal{A}}$, along with the key k.

- $\mathcal{R}_{\mathcal{B}}$ (respectively, $\mathcal{R}_{\mathcal{C}}$) on receiving $(\sigma_{\mathcal{B}}, k)$ (respectively, $(\sigma_{\mathcal{C}}, k)$) from $\mathcal{R}_{\mathcal{A}}$ and $x^{\mathcal{B}}$ (respectively, $x^{\mathcal{C}}$) from Ch computes $y^{\mathcal{B}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{B}})$ (respectively, $y^{\mathcal{C}} \leftarrow \mathsf{PRF.Eval}(k, x^{\mathcal{C}})$) and $\mathsf{ct}^{\mathcal{B}} = (x^{\mathcal{B}}, y^{\mathcal{B}} \oplus m_0^{\mathcal{B}})$ (respectively, $\mathsf{ct}^{\mathcal{C}} = (x^{\mathcal{C}}, y^{\mathcal{C}} \oplus m_0^{\mathcal{C}})$) and runs $\mathcal{B}$ on $\mathsf{ct}^{\mathcal{B}}$ (respectively, $\mathcal{C}$ on $\mathsf{ct}^{\mathcal{C}}$) to get a bit $b^{\mathcal{B}}$ (respectively, $b^{\mathcal{C}}$), and outputs $b^{\mathcal{B}}$ (respectively, $b^{\mathcal{C}}$).

□

Remark 67. *If we change the* UPO *security guarantee of the underlying* UPO *scheme from $\mathcal{U}$-generalized* UPO *security to $\mathsf{Id}_{\mathcal{U}}$-generalized* UPO *security (see Section 3.1.1), then using the same proof as in Proposition 66 upto minor corrections, we achieve $\mathcal{D}_{\mathsf{identical}}$-selective* CPA *anti-piracy instead of $\mathcal{D}_{\mathsf{iden\text{-}bit,ind\text{-}msg}}$-selective* CPA *anti-piracy as in Proposition 66 for the* SDE *scheme given in Figure 20.*

Theorem 68 (SDE lifting theorem)**.** *Assuming post-quantum indistinguishability obfuscation for classical circuits and length-doubling injective pseudorandom generators, there is a generic lift that takes a $\mathcal{D}_{\mathsf{iden\text{-}bit,ind\text{-}msg}}$-selective* CPA *secure* SDE *scheme and outputs a new* SDE *that is full-blown $\mathcal{D}_{\mathsf{iden\text{-}bit,ind\text{-}msg}}$-*CPA *secure (see Appendix A.2).*

Proof. Let $(\mathsf{Gen}, \mathsf{QKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a selectively CPA secure SDE, and let iO be an indistinguishability obfuscation. Consider the SDE scheme $(\mathsf{Gen}', \mathsf{QKeyGen}', \mathsf{Enc}', \mathsf{Dec}')$ given in Figure 25.

Assumes: SDE scheme $(\mathsf{Gen}, \mathsf{QKeyGen}, \mathsf{Enc}, \mathsf{Dec})$, post-quantum indistinguishability obfuscation iO.

$\mathsf{Gen}'(1^\lambda)$: Same as $\mathsf{Gen}()$.

$\mathsf{QKeyGen}'(\mathsf{sk})$: Same as $\mathsf{QKeyGen}()$.

$\mathsf{Enc}'(\mathsf{pk}, m)$:

1. Sample $r \xleftarrow{\$} \{0,1\}^n$.
2. Generate $c = \mathsf{Enc}(\mathsf{pk}, r)$.
3. Output $\mathsf{ct} = (\tilde{C}, c)$, where $\tilde{C} \leftarrow \mathsf{iO}(C)$ and C is the circuit that on input r outputs m and outputs $\perp$ on all other inputs.

$\mathsf{Dec}'(\rho_{\mathsf{sk}}, \mathsf{ct})$

1. Interprete $\mathsf{ct} = \tilde{C}, c$.
2. Run $r \leftarrow \mathsf{Dec}(\rho_{\mathsf{sk}}, c)$.
3. Output $m = \tilde{C}(r)$.

Figure 25: A construction of CPA-secure single decryptor encryption from a selectively CPA-secure single decryptor encryption.

The correctness of $(\mathsf{Gen}', \mathsf{QKeyGen}', \mathsf{Enc}', \mathsf{Dec}')$ follows directly from the correctness of $(\mathsf{Gen}, \mathsf{QKeyGen}, \mathsf{Enc}, \mathsf{Dec})$.

CPA anti-piracy of $(\mathsf{Gen}', \mathsf{QKeyGen}', \mathsf{Enc}', \mathsf{Dec}')$ from selective security of $(\mathsf{Gen}, \mathsf{QKeyGen}, \mathsf{Enc}, \mathsf{Dec})$. Let $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ be an adversary against the full-blown CPA security experiment for the $\mathsf{CPA.SDE.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C})}(1^\lambda)$ (see Figure 36). We will do a sequence of hybrids starting from the original anti-piracy experiment $\mathsf{CPA.SDE.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C})}(1^\lambda)$ for the single decryptor encryption scheme given in Figure 25, and then conclude with a reduction from the final to. The changes are marked in blue.

Hybrid_0:

Same as $\mathsf{CPA.SDE.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C})}(1^\lambda)$ given in Figure 36 for the single decryptor encryption scheme in Figure 25.

- Ch samples $\mathsf{sk}, \mathsf{pk} \leftarrow \mathsf{Gen}(1^\lambda)$ and generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{QKeyGen}(\mathsf{sk})$ and sends $(\rho_{\mathsf{sk}}, \mathsf{pk})$ to $\mathcal{A}$.
- $\mathcal{A}$ sends two messages $(m_0^{\mathcal{B}}, m_1^{\mathcal{B}}, m_0^{\mathcal{C}}, m_1^{\mathcal{C}})$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- Ch samples $b \xleftarrow{\$} \{0,1\}$ and generates $c^{\mathcal{B}} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b^{\mathcal{B}})$ and $c^{\mathcal{C}} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b^{\mathcal{C}})$.

- $\mathcal{A}$ samples $r_0^{\mathcal{B}}, r_1^{\mathcal{B}}, r_0^{\mathcal{C}}, r_1^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$ and computes $\mathsf{ct}^{\mathcal{B}} = (\mathsf{iO}(C^{\mathcal{B}}), c^{\mathcal{B}})$ and $\mathsf{ct}^{\mathcal{C}} = (\mathsf{iO}(C^{\mathcal{C}}), c^{\mathcal{C}})$, where $C^{\mathcal{B}}$ and $C^{\mathcal{C}}$ are the circuits that on input $r_b^{\mathcal{B}}$ and $r_b^{\mathcal{C}}$ respectively, outputs $m_b^{\mathcal{B}}$ and $m_b^{\mathcal{C}}$, respectively. $C^{\mathcal{B}}$ and $C^{\mathcal{C}}$ on all inputs except $r_b^{\mathcal{B}}$ and $r_b^{\mathcal{C}}$ respectively, outputs $\perp$.
- Apply $(\mathcal{B}(\mathsf{ct}^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.
- Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$.

Hybrid_1:

- Ch samples $\mathsf{sk}, \mathsf{pk} \leftarrow \mathsf{Gen}(1^\lambda)$ and generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{QKeyGen}(\mathsf{sk})$ and sends $(\rho_{\mathsf{sk}}, \mathsf{pk})$ to $\mathcal{A}$.
- $\mathcal{A}$ sends two messages $(m_0^{\mathcal{B}}, m_1^{\mathcal{B}}, m_0^{\mathcal{C}}, m_1^{\mathcal{C}})$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- Ch samples $b \xleftarrow{\$} \{0,1\}$ and generates $c^{\mathcal{B}} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b^{\mathcal{B}})$ and $c^{\mathcal{C}} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b^{\mathcal{C}})$.
- $\mathcal{A}$ samples $r_0^{\mathcal{B}}, r_1^{\mathcal{B}}, r_0^{\mathcal{C}}, r_1^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$, $y^{\mathcal{B}}, y^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$, and computes $\mathsf{ct}^{\mathcal{B}} = (\mathsf{iO}(C^{\mathcal{B}}), c^{\mathcal{B}})$ and $\mathsf{ct}^{\mathcal{C}} = (\mathsf{iO}(C^{\mathcal{C}}), c^{\mathcal{C}})$, where $C^{\mathcal{B}}$ and $C^{\mathcal{C}}$ are the circuits ~~that on input $r^{\mathcal{B}}$ and $r^{\mathcal{C}}$ respectively, outputs $m^{\mathcal{B}}$ and $m^{\mathcal{C}}$, respectively. $C^{\mathcal{B}}$ and $C^{\mathcal{C}}$ on all inputs except $r^{\mathcal{B}}$ and $r^{\mathcal{C}}$ respectively, outputs $\perp$.~~ are as depicted in Figures 26 and 27, respectively.
- Apply $(\mathcal{B}(\mathsf{ct}^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.
- Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$.

The indistinguishability between hybrids Hybrid_0 and Hybrid_1 holds because of the following. Since the PRG is a length-doubling, except with negligible probability, the functionality of circuits $C^{\mathcal{B}}$ and $C^{\mathcal{C}}$ did not change across the hybrids Hybrid_0 and Hybrid_1. Therefore the computational indistinguishability between Hybrid_0 and Hybrid_1 follows from the security guarantees of iO.

<u>$C^{\mathcal{B}}$</u>:

Hardcoded keys $r_b^{\mathcal{B}}, m_b^{\mathcal{B}}, m_{1-b}^{\mathcal{B}}, y^{\mathcal{B}}$. On input: r.

- If $r = r_b^{\mathcal{B}}$, output $m_b^{\mathcal{B}}$.
- If $\mathsf{PRG}(r) = y^{\mathcal{B}}$, output $m_{1-b}^{\mathcal{B}}$.
- Otherwise, output $\perp$.

Figure 26: Circuit $C^{\mathcal{B}}$ in Hybrid_1

Hybrid_2:

$\underline{C^{\mathcal{C}}}$:

Hardcoded keys $r_b^{\mathcal{C}}, m_b^{\mathcal{C}}, m_{1-b}^{\mathcal{B}}, y^{\mathcal{B}}$. On input: r.

- If $r = r_b^{\mathcal{C}}$, output $m_b^{\mathcal{C}}$.
- If $\mathsf{PRG}(r) = y^{\mathcal{C}}$, output $m_{1-b}^{\mathcal{C}}$.
- Otherwise, output $\perp$.

Figure 27: Circuit $C^{\mathcal{C}}$ in Hybrid_1

- Ch samples $\mathsf{sk}, \mathsf{pk} \leftarrow \mathsf{Gen}(1^\lambda)$ and generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{QKeyGen}(\mathsf{sk})$ and sends $(\rho_{\mathsf{sk}}, \mathsf{pk})$ to $\mathcal{A}$.
- $\mathcal{A}$ sends two messages $(m_0^{\mathcal{B}}, m_1^{\mathcal{B}}, m_0^{\mathcal{C}}, m_1^{\mathcal{C}})$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- Ch samples $b \xleftarrow{\$} \{0,1\}$ and generates $c^{\mathcal{B}} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b^{\mathcal{B}})$ and $c^{\mathcal{C}} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b^{\mathcal{C}})$.
- $\mathcal{A}$ samples $r^{\mathcal{B}}, r^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$, ~~$y^{\mathcal{B}}, y^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$~~, $y^{\mathcal{B}} \leftarrow \mathsf{PRG}(r_{1-b}^{\mathcal{B}})$, $y^{\mathcal{C}} \leftarrow \mathsf{PRG}(r^{\mathcal{C}}1-b)$ and computes $\mathsf{ct}^{\mathcal{B}} = (\mathsf{iO}(C^{\mathcal{B}}), c^{\mathcal{B}})$ and $\mathsf{ct}^{\mathcal{C}} = (\mathsf{iO}(C^{\mathcal{C}}), c^{\mathcal{C}})$, where $C^{\mathcal{B}}$ and $C^{\mathcal{C}}$ are the circuits are as depicted in Figures 26 and 27, respectively.
- Apply $(\mathcal{B}(\mathsf{ct}^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.
- Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$.

The indistinguishability between Hybrid_1 and Hybrid_2 holds due to pseudorandomness of PRG.
Hybrid_3:

- Ch samples $\mathsf{sk}, \mathsf{pk} \leftarrow \mathsf{Gen}(1^\lambda)$ and generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{QKeyGen}(\mathsf{sk})$ and sends $(\rho_{\mathsf{sk}}, \mathsf{pk})$ to $\mathcal{A}$.
- $\mathcal{A}$ sends two messages $(m_0^{\mathcal{B}}, m_1^{\mathcal{B}}, m_0^{\mathcal{C}}, m_1^{\mathcal{C}})$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- Ch samples $b \xleftarrow{\$} \{0,1\}$ and generates $c^{\mathcal{B}} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b^{\mathcal{B}})$ and $c^{\mathcal{C}} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b^{\mathcal{C}})$.
- $\mathcal{A}$ samples $r^{\mathcal{B}}, r^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$, ~~$y^{\mathcal{B}} \leftarrow \mathsf{PRG}(r_{1-b}^{\mathcal{B}})$, $y^{\mathcal{C}} \leftarrow \mathsf{PRG}(r^{\mathcal{C}}1-b)$~~ and computes $\mathsf{ct}^{\mathcal{B}} = (\mathsf{iO}(C^{\mathcal{B}}), c^{\mathcal{B}})$ and $\mathsf{ct}^{\mathcal{C}} = (\mathsf{iO}(C^{\mathcal{C}}), c^{\mathcal{C}})$, where $C^{\mathcal{B}}$ and $C^{\mathcal{C}}$ are the circuits are as depicted in Figures 26 and 27, respectively.
- Apply $(\mathcal{B}(\mathsf{ct}^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.
- Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$.

The indistinguishability between Hybrid_2 and Hybrid_3 holds immediately by the iO guarantees since we did not change the functionality of $C^{\mathcal{B}}$ and $C^{\mathcal{C}}$ across the hybrids Hybrid_2 and Hybrid_3.

$\underline{C^{\mathcal{B}}}$:

Hardcoded keys $r_b^{\mathcal{B}}, m_0^{\mathcal{B}}, m_1^{\mathcal{B}}, \sout{y^{\mathcal{B}}}, r_{1-b}^{\mathcal{B}}$. On input: r.

- If $r = r_b^{\mathcal{B}}$, output $m_b^{\mathcal{B}}$.
- ~~If $\mathsf{PRG}(r) = y^{\mathcal{B}}$, output $m_{1-b}^{\mathcal{B}}$.~~ If $r = r_{1-b}^{\mathcal{B}}$, output $m_{1-b}^{\mathcal{B}}$.
- Otherwise, output $\perp$.

Figure 28: Circuit $C^{\mathcal{B}}$ in Hybrid_3

$\underline{C^{\mathcal{C}}}$:

Hardcoded keys $r_b^{\mathcal{C}}, m^{\mathcal{B}}, m^{\mathcal{C}}, \sout{y^{\mathcal{C}}} r_{1-b}^{\mathcal{C}}$. On input: r.

- If $r = r_b^{\mathcal{C}}$, output $m_b^{\mathcal{C}}$.
- ~~If $\mathsf{PRG}(r) = y^{\mathcal{C}}$, output $m_{1-b}^{\mathcal{C}}$.~~ If $r = r_{1-b}^{\mathcal{C}}$, output $m_{1-b}^{\mathcal{C}}$.
- Otherwise, output $\perp$.

Figure 29: Circuit $C^{\mathcal{C}}$ in Hybrid_3

Finally we give a reduction from Hybrid_3 to the selective-CPA anti-piracy game for $(\mathsf{Gen}, \mathsf{QKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ given in Figure 35. Let $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ be an adversary in Hybrid_3 above. Consider the following non-local adversary $(\mathcal{R}_{\mathcal{A}}, \mathcal{R}_{\mathcal{B}}, \mathcal{R}_{\mathcal{C}})$:

- $\mathcal{R}_{\mathcal{A}}$ samples $r_0^{\mathcal{B}}, r_1^{\mathcal{B}}, r_0^{\mathcal{C}}, r_1^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^n$, and sends $(r_0^{\mathcal{B}}, r_1^{\mathcal{B}})$ and $(r_0^{\mathcal{C}}, r_1^{\mathcal{C}})$ as the challenge messages to Ch, the challenger for the selective-CPA anti-piracy game for $(\mathsf{Gen}, \mathsf{QKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ given in Figure 35.
- $\mathcal{R}_{\mathcal{A}}$ on receiving the decryptor and the public key (ρ, pk) from Ch runs $\mathcal{A}$ on (ρ, pk) to gets back the output, two pairs of messages $(m_0^{\mathcal{B}}, m_1^{\mathcal{B}})$ and $(m_0^{\mathcal{C}}, m_1^{\mathcal{C}})$ and a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- $\mathcal{R}_{\mathcal{A}}$ constructs the circuit $\mathsf{iO}(C^{\mathcal{B}})$ and $\mathsf{iO}(C^{\mathcal{C}})$ where $C^{\mathcal{B}}$ and $C^{\mathcal{C}}$ are the circuits are as depicted in Figures 26 and 27, respectively.
- $\mathcal{R}_{\mathcal{A}}$ sends $\mathsf{iO}(C^{\mathcal{B}}), \sigma_{\mathcal{B}}$ to $\mathcal{R}_{\mathcal{B}}$ and $\mathsf{iO}(C^{\mathcal{C}}), \sigma_{\mathcal{C}}$ to $\mathcal{R}_{\mathcal{C}}$.

- $\mathcal{R}_{\mathcal{B}}$ on receiving $c^{\mathcal{B}}$ from Ch and $(\mathsf{iO}(C^{\mathcal{B}}), \sigma_{\mathcal{B}})$ from $\mathcal{R}_{\mathcal{A}}$, runs $b^{\mathcal{B}} \leftarrow \mathcal{B}(\sigma_{\mathcal{B}}, (\mathsf{iO}(C^{\mathcal{B}}), c^{\mathcal{B}}))$ and outputs $b^{\mathcal{B}}$.
- $\mathcal{R}_{\mathcal{C}}$ on receiving $c^{\mathcal{C}}$ from Ch and $(\mathsf{iO}(C^{\mathcal{C}}), \sigma_{\mathcal{C}})$ from $\mathcal{R}_{\mathcal{A}}$, runs $b^{\mathcal{C}} \leftarrow \mathcal{C}(\sigma_{\mathcal{C}}, (\mathsf{iO}(C^{\mathcal{C}}), c^{\mathcal{C}}))$ and outputs $b^{\mathcal{C}}$.

$\underline{C^{\mathcal{B}}}$:

Hardcoded keys $r_b^{\mathcal{B}}, m_0^{\mathcal{B}}, m_1^{\mathcal{B}}, \cancel{y^{\mathcal{B}}}, r_{1-b}^{\mathcal{B}}$. On input: r.

- If $r = r_b^{\mathcal{B}}$, output $m_b^{\mathcal{B}}$.
- If $r = r_{1-b}^{\mathcal{B}}$, output $m_{1-b}^{\mathcal{B}}$.
- Otherwise, output $\perp$.

Figure 30: Circuit $C^{\mathcal{B}}$

$\underline{C^{\mathcal{C}}}$:

Hardcoded keys $r_b^{\mathcal{C}}, m^{\mathcal{B}}, m^{\mathcal{C}}, \cancel{y^{\mathcal{C}}} r_{1-b}^{\mathcal{C}}$. On input: r.

- If $r = r_b^{\mathcal{C}}$, output $m_b^{\mathcal{C}}$.
- If $r = r_{1-b}^{\mathcal{C}}$, output $m_{1-b}^{\mathcal{C}}$.
- Otherwise, output $\perp$.

Figure 31: Circuit $C^{\mathcal{C}}$

$\square$

Remark 69. *The proof of theorem 68 can be adapted to prove the same construction lifts a* SDE *with* $\mathcal{D}_{\mathsf{identical}}$*-selective* CPA *anti-piracy to* $\mathcal{D}_{\mathsf{identical}}$-CPA *anti-piracy.*

Remarks 67 and 69 together gives us the following corollary.

Corollary 70. *Assuming an indistinguishability obfuscation scheme* iO *for* P/poly*, a puncturable pseudorandom function family* PRF = (Gen, Eval, Puncture)*, and a* $\mathsf{Id}_{\mathcal{U}}$*-generalized* UPO *scheme for any generalized puncturable keyed circuit class in* P/poly *(see Section 3.1.1 for the formal definition of* $\mathsf{Id}_{\mathcal{U}}$*), there exists a secure public-key unclonable encryption for multiple bits.*

Proof. By [GZ20], a SDE scheme for multiple-bit messages satisfying $\mathcal{D}_{\mathsf{identical}}$-selective CPA anti-piracy, implies private-key unclonable encryption for multiple bits. Then the result of [AK21] shows that there exists a transformation from one-time unclonable encryption to public-key unclonable encryption assuming post-quantum secure public-key encryption, which in turn can be instantiated using iO and puncturable pseudorandom functions [SW14]. □

6.5 Copy-Protection for Evasive Functions

We will be considering the following class of evasive function classes.

Definition 71. *A class of keyed boolean-valued functions with input-length $n = n(\lambda)$ $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ is evasive with respect to an efficiently samplable distribution $\mathcal{D}_\mathcal{F}$ on $\mathcal{F}$, if for every fixed input point x, there exists a negligible function* negl() *such that*

$$\Pr[f \leftarrow \mathcal{D}_\mathcal{F}(1^\lambda) : f(x) = 1] = \mathsf{negl}(\lambda).$$

Definition 72 (preimage-samplable evasive functions)**.** *An evasive function class $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ is* preimage-samplable *equipped with a distribution $\mathcal{D}_\mathcal{F}$ on $\mathcal{K}$ is a* preimage-samplable *evasive function class if*

1. *There exists a keyed circuit implementation $(\mathcal{D}, \mathfrak{C}^\mathcal{F})$ of $(\mathcal{D}_\mathcal{F}, \mathcal{F})$ where $\mathfrak{C}^\mathcal{F} = \{C_k^\mathcal{F}\}_{k\in\mathcal{K}}$.*
2. *There exists an auxiliary generalized puncturable keyed circuit class $\mathfrak{C} = \{C_{k'}\}_{k'\in\mathcal{K}'}$ with* Evasive-GenPuncture *as the generalized puncturing algorithm, (see Section 3.1.1), and equipped with an efficiently samplamable distribution $\mathcal{D}'$ on its keyspace $\mathcal{K}'$,*

such that

$$\{C^{\mathcal{F}}{}_k, x\}_{k\leftarrow\mathcal{D}(1^\lambda), x\xleftarrow{\$}C_k^{\mathcal{F}^{-1}}(1)} \approx_c \{C_{k',y,\vec{1}}, y\}_{C_{k',y,\vec{1}}\leftarrow\mathsf{Evasive\text{-}GenPuncture}(k',y,y,\vec{1},\vec{1}), k'\leftarrow\mathcal{D}'(1^\lambda), y\xleftarrow{\$}\{0,1\}^n}, \quad (8)$$

where $\vec{1}$ is the constant-1 function, and $C_{k',y,\vec{1}}$ is the same as the circuit $C_{k',y,y,\vec{1},\vec{1}}$. In short, we call $(\mathcal{D}_\mathcal{F}, \mathcal{F})$ preimage-samplable *evasive if* preimage-samplable *equipped with a distribution $\mathcal{D}_\mathcal{F}$ on $\mathcal{K}$ is a* preimage-samplable *evasive function class.*

Instantiations:

Theorem 73. *For every $t \in [2^n]$, let $\mathcal{F}^t = \{\mathcal{F}_\lambda^t\}$ defined as $\mathcal{F}_\lambda^t = \{f : \{0,1\}^n \mapsto \{0,1\} \mid |f^{-1}(1)| = t\}$, i.e, the set of all functions f on n-bit input with exactly t non-zero input points. Suppose r is such that the following holds:*

1. *$\mathcal{F}^r$ is evasive with respect to $\mathcal{U}_{\mathcal{F}^r}$, the uniform distribution.*
2. *There exists a keyed circuit implementation $(\mathcal{D}^r, \mathfrak{C}^r)$ for $(\mathcal{U}_{\mathcal{F}^r}, \mathcal{F}^r)$, and similarly keyed circuit implementation $(\mathcal{D}^{r-1}, \mathfrak{C}^{r-1})$ for $(\mathcal{U}_{\mathcal{F}^{r-1}}, \mathcal{F}^{r-1})$.*

Then, assuming post-quantum indistinguishability obfuscation, $(\mathcal{U}_{\mathcal{F}^r}, \mathcal{F}^r)$ is preimage-samplable *evasive.*

Proof. Let $r \in o(2^n)$ as given in the theorem. Fix the circuit descriptions $\mathfrak{C}^r$ and $\mathfrak{C}^{r-1}$ for $\mathcal{F}^{r-1}$ and $\mathcal{F}^{r-1}$ respectively, as mentioned in the theorem.

Note that for every circuit $k \in \mathcal{K}_\lambda^{r-1}$ and set of inputs $\{x_1, x_2\}$ and circuits $\{\mu_1, \mu_2\}$, there is an efficient procedure to construct the circuit $C_{k,x_1,x_2,\mu_1,\mu_2}$ which on any input x' first checks if $x' = x_i$ for some $i \in [2]$ in which case it outputs $\mu_i(x_i)$, otherwise it outputs $C_k^{r-1}(x)$. We call this procedure $\mathsf{GenPuncture}$. For $x_1 = x_2 = y$ and $\mu_1 = \mu_2 = \mu$, we will use $C_{k,y,\mu}$ as a shorthand notation for $C_{k,x_1,x_2,\mu_1,\mu_2}$.

We assume that for every $\lambda \in \mathbb{N}$, and for every $k \in \mathcal{K}_\lambda^r$, and for every $k' \in \mathcal{K}_\lambda^{r-1}$, and $x_1, x_2 \in \{0,1\}^n$, circuit $C_k^r \in \mathfrak{C}^r$, $C_{k'}^{r-1} \in \mathfrak{C}^{r-1}$, and a punctured circuit $C_{k',x_1,x_2,\mu_1,\mu_2} \leftarrow \mathsf{GenPuncture}(k', x_1, x_2, \mu_1, \mu_2)$ have the same size. These conditions can be achieved by padding sufficiently many zeroes to smaller circuits.

Let iO be a post-quantum indistinguishability obfuscation.

Next, we make the following claim

Claim 74.

$$\{\mathsf{iO}(C_k^r), x\}_{k \leftarrow \mathcal{D}^r(1^\lambda), x \xleftarrow{\$} C_k^{r-1}(1)} \approx_c \{\mathsf{iO}(C_{k',y,\vec{1}}), y\}_{k' \leftarrow \mathcal{D}^{r-1}(1^\lambda), y \xleftarrow{\$} \{0,1\}^n}.$$

We first prove the theorem assuming Claim 74 as follows. Let $a(\lambda)$ be the amount of randomness iO uses to obfuscate the circuits in $\mathfrak{C}^r$ and the punctured circuits obtained by puncturing circuits in $\mathfrak{C}^{r-1}$ using the $\mathsf{GenPuncture}$ algorithm.

Fix a security parameter λ arbitrarily.

Let $\widetilde{\mathfrak{C}}^r = \{\{\mathsf{iO}(C_k^r; t)\}_{k \in \mathcal{K}_\lambda^r, t \in \{0,1\}^{a(\lambda)}}\}_\lambda$ be a keyed circuit class with keyspace $\mathcal{K}^r \times \{0,1\}^a$. Note that by the correctness of iO, for every $k \in \mathcal{K}_\lambda^r$, the circuit $\mathsf{iO}(C_k^r; t)$ has the same functionality as C_k^r for every $t \in \{0,1\}^{a(\lambda)}$, i.e, $S_\lambda(\mathsf{iO}(C_k^r; t)) = S_\lambda(C_k^r)$ where S_λ is the canonical circuit-to-functionality map. Therefore, since $\mathfrak{C}^r$ is a keyed implementation $\mathcal{F}^r$, so is $\widetilde{\mathfrak{C}}^r$ (see Section 6.1 for the definition of keyed implementation). Moreover, since $S_\lambda(\mathsf{iO}(C_k^r; t)) = S_\lambda(C_k^r)$, it holds that

$$\{S_\lambda(C_k^r)\}_{k \leftarrow \mathcal{D}^r(1^\lambda)} = \{S_\lambda(\mathsf{iO}(C_k^r; t))\}_{k \leftarrow \mathcal{D}^r(1^\lambda), t \xleftarrow{\$} \{0,1\}^{a(\lambda)}}.$$

Therefore, since $(\mathcal{D}^r, \mathfrak{C}^r)$ is a keyed implementation of $(\mathcal{U}_{\mathcal{F}^r}, \mathcal{F}^r)$, so is $(\mathcal{D}, \widetilde{\mathfrak{C}}^r)$ where $\mathcal{D}$ is defined as $(k, t) \leftarrow \mathcal{D}(1^\lambda) \equiv k \leftarrow \mathcal{D}^r(1^\lambda), t \xleftarrow{\$} \{0,1\}^{a(\lambda)}$ (see Section 6.1 for the definition of keyed implementation).

Similarly, $(\mathcal{D}', \widetilde{\mathfrak{C}}^{r-1})$ is a generalized circuit implementation of $(\mathcal{U}_{\mathcal{F}^{r-1}}, \mathcal{F}^{r-1})$ where $\mathcal{D}'$ is defined as $(k, t) \leftarrow \mathcal{D}'(1^\lambda) \equiv k \leftarrow \mathcal{D}^{r-1}(1^\lambda), t \xleftarrow{\$} \{0,1\}^{a(\lambda)}$ and $\widetilde{\mathfrak{C}}^{r-1} = \{\{\mathsf{iO}(C_k^{r-1}; t)\}_{k \in \mathcal{K}_\lambda^{r-1}, t \in \{0,1\}^{a(\lambda)}}\}_\lambda$.

Let $\mathsf{Evasive\text{-}GenPuncture}$ be an efficient algorithm that on input $k' \in \mathcal{K}_\lambda^{r-1}$, $t' \in \{0,1\}^a$, a set of points y_1, y_2 and circuits μ_1, μ_2, generates $C_{k',y_1,y_2,\mu_1,\mu_2}$ and outputs the circuit $\mathsf{iO}(C_{k',y_1,y_2,\mu_1,\mu_2}; t')$.

Note that by definition of $\mathcal{D}$,

$$\{\mathsf{iO}(C_k^r; t), x\}_{(k,t) \leftarrow \mathcal{D}(1^\lambda) x \xleftarrow{\$} \{C_k^r\}^{-1}(1)} = \{\mathsf{iO}(C_k^r), x\}_{k \leftarrow \mathcal{D}^r(1^\lambda), x \xleftarrow{\$} \{C_k^r\}^{-1}(1)},$$

which is the LHS of Claim 74, and,

$$\begin{aligned}
&\{\tilde{C}_{k',t',y',\vec{1}}, y\}_{\tilde{C}_{k',t',y',\vec{1}} \leftarrow \mathsf{Evasive\text{-}GenPuncture}((k',t'),y,y,\vec{1},\vec{1}), (k',t') \leftarrow \mathcal{D}'(1^\lambda), y \xleftarrow{\$} \{0,1\}^n} \\
&= \{\mathsf{iO}(C_{k',y,\vec{1}}; t'), y\}_{C_{k',y,\vec{1}} \leftarrow \mathsf{GenPuncture}(k',y,y,\vec{1},\vec{1}), (k',t') \leftarrow \mathcal{D}'(1^\lambda), y \xleftarrow{\$} \{0,1\}^n} && \text{By definition of } \mathsf{Evasive\text{-}GenPuncture} \\
&= \{\mathsf{iO}(C_{k',y,\vec{1}}), y\}_{k' \leftarrow \mathcal{D}^{r-1}(1^\lambda), y \xleftarrow{\$} \{0,1\}^n}, && \text{By definition of } \mathcal{D}'
\end{aligned}$$

which is the RHS of Claim 74. Hence by Claim 74, we conclude that,

$$\{\mathsf{iO}(C_k^r;t),x\}_{k,t\leftarrow\mathcal{D}(1^\lambda)),x\xleftarrow{\$}\{C_k^r\}^{-1}(1)}$$
$$\approx_c \{\tilde{C}_{k',t',y',\vec{1}},y\}_{\tilde{C}_{k',t',y',\vec{1}}\leftarrow\mathsf{Evasive\text{-}GenPuncture}(k',y,y,\vec{1},\vec{1}),k'\leftarrow\mathcal{D}'(1^\lambda),t'\xleftarrow{\$}\{0,1\}^a,y\xleftarrow{\$}\{0,1\}^n},$$

which is exactly the preimage-samplable condition for $\mathcal{U}_{\mathcal{F}^r},\mathcal{F}^r$ with the keyed circuit implementation, $(\mathcal{D},\widetilde{\mathfrak{C}}^r)$, the auxiliary generalized puncturable keyed circuit class $\widetilde{\mathfrak{C}}^{r-1}$ equipped with Evasive-GenPuncture, and $\mathcal{D}'$ as the corresponding distribution on the keyspace of $\widetilde{\mathfrak{C}}^{r-1}$.

Next, we give a proof of Claim 74 to complete the proof.

Proof of Claim 74 Fix λ arbitrarily. Since $\mathcal{F}^r$ is evasive, so is $\mathcal{F}^{r-1}$. Hence, $k'\leftarrow\mathcal{D}^{r-1}(1^\lambda)$, $y\xleftarrow{\$}\{0,1\}^n\approx_s y\xleftarrow{\$}\{C_{k'}^{r-1}\}^{-1}(0)$ and hence,

$$\{\mathsf{iO}(C_{k',y,\vec{1}}),y\}_{k'\leftarrow\mathcal{D}^{r-1}(1^\lambda),y\xleftarrow{\$}\{0,1\}^n}\approx_s\{\mathsf{iO}(C_{k,y,\vec{1}}),y\}_{k\xleftarrow{\$}\mathcal{K}_\lambda^{r-1},y\xleftarrow{\$}\{C_{k'}^{r-1}\}^{-1}(0)}.$$

Hence it is enough to show that

$$\{\mathsf{iO}(C_k^r),x\}_{k\leftarrow\mathcal{D}^r(1^\lambda),x\xleftarrow{\$}C_k^{r-1}(1)}\approx_c\{\mathsf{iO}(C_{k',y,\vec{1}}),y\}_{k'\leftarrow\mathcal{D}^{r-1}(1^\lambda),y\xleftarrow{\$}\{C_{k'}^{r-1}\}^{-1}(0)}.$$

Recall the circuit-to-functionality map S_λ. Let Induced-$\mathcal{D}^r$ and Induced-$\mathcal{D}^{r-1}$ be the distribution that $\mathcal{D}^r$ and $\mathcal{D}^{r-1}$ respectively induces on $\mathcal{F}_\lambda^r$ and $\mathcal{F}_\lambda^{r-1}$ under S_λ. Since $(\mathcal{D}^r,\mathfrak{C}^r)$ and $(\mathcal{D}^{r-1},\mathfrak{C}^{r-1})$ are keyed implementation of $(\mathcal{U}_{\mathcal{F}^r},\mathcal{F}^r)$ and $(\mathcal{U}_{\mathcal{F}^{r-1}},\mathcal{F}^{r-1})$ respectively, it holds that,

$$\mathsf{Induced\text{-}}\mathcal{D}^r\approx_s\mathcal{U}_{\mathcal{F}^r},\text{ and similarly, }\mathsf{Induced\text{-}}\mathcal{D}^{r-1}\approx_s\mathcal{U}_{\mathcal{F}^{r-1}}\tag{9}$$

Since $\widetilde{\mathfrak{C}}^r$ and $\widetilde{\mathfrak{C}}^{r-1}$ are keyed implementations of $\mathcal{F}^r$ and $\mathcal{F}^{r-1}$ respectively, for every $f\in\mathcal{F}^r$ and $\mathcal{G}^{r-1}$ $\mathcal{D}^r$ and $\mathcal{D}^{r-1}$ induce distributions $\mathcal{D}^r$-S_f and $\mathcal{D}^{r-1}$-S_g, on the class of circuits $S_\lambda^{-1}(f)$ and $S_\lambda^{-1}(g)$, respectively. For every $f\in\mathcal{F}^r,g\in\mathcal{F}^{r-1}$, let k_f and k'_g be the lexicographically first key in $\mathcal{K}^r$ and $\mathcal{K}^{r-1}$ such that $C_{k_f}^r\in S_\lambda^{-1}(f)$ and $C_{k'_g}^{r-1}\in S_\lambda^{-1}(g)$.

Note that by the security of iO, for every $f\in\mathcal{F}^r$, and $C_k^r\in S_\lambda^{-1}(f)$

$$\{\mathsf{iO}(C_k^r;t)\}_{t\xleftarrow{\$}\{0,1\}^a}\approx_c\{\mathsf{iO}(C_{k_f}^r;t)\}_{t\xleftarrow{\$}\{0,1\}^a}.$$

Therefore it holds that, for every $f\in\mathcal{F}^r$,

$$\{\mathsf{iO}(C_k^r)\}_{k\leftarrow\mathcal{D}^r\text{-}S_f}=\{\mathsf{iO}(C_k^r;t)\}_{k\leftarrow\mathcal{D}^r\text{-}S_f,t\xleftarrow{\$}\{0,1\}^a}\approx_c\{\mathsf{iO}(C_{k_f}^r;t)\}_{t\xleftarrow{\$}\{0,1\}^a}=\{\mathsf{iO}(C_{k_f}^r)\}.\tag{10}$$

Next note that,

$$\{\mathsf{iO}(C_k^r),x\}_{k\leftarrow\mathcal{D}^r(1^\lambda),x\xleftarrow{\$}\{C_k^r\}^{-1}(1)}=\{\mathsf{iO}(C_k^r),x\}_{k\leftarrow\mathcal{D}^r\text{-}S_f(1^\lambda),f\leftarrow\mathsf{Induced\text{-}}\mathcal{D}^r x\xleftarrow{\$}C_k^{r-1}(1)}.$$

Therefore,

$$\begin{aligned}
&\{\mathsf{iO}(C_k^r), x\}_{k\leftarrow\mathcal{D}^r(1^\lambda), x\xleftarrow{\$}\{C_k^r\}^{-1}(1)} \\
&= \{\mathsf{iO}(C_k^r), x\}_{k\leftarrow\mathcal{D}^r\text{-}S_f(1^\lambda), f\leftarrow\mathsf{Induced}\text{-}\mathcal{D}^r, x\xleftarrow{\$}\{\mathsf{iO}(C_k^r)\}^{-1}(1)} \\
&\approx_s \{\mathsf{iO}(C_k^r), x\}_{k\leftarrow\mathcal{D}^r\text{-}S_f(1^\lambda), f\leftarrow\mathcal{U}_{\mathcal{F}^r}, x\xleftarrow{\$}C_k^{r\,-1}(1)} && \text{By Equation (9)} \\
&\approx_c \{\mathsf{iO}(C_{k_f}^r; t), x\}_{t\xleftarrow{\$}\{0,1\}^a, f\leftarrow\mathcal{U}_{\mathcal{F}^r}, x\xleftarrow{\$}\{\mathsf{iO}(C_{k_f}^r)\}^{-1}(1)} && \text{By Equation (10)} \\
&= \{\mathsf{iO}(C_{k_f}^r; t), x\}_{t\xleftarrow{\$}\{0,1\}^a, f\leftarrow\mathcal{U}_{\mathcal{F}^r}, x\xleftarrow{\$}f^{-1}(1)}.
\end{aligned}$$

Similarly, it can be shown that

$$\{\mathsf{iO}(C_{k',y,\vec{1}}), y\}_{k'\leftarrow\mathcal{D}^{r-1}(1^\lambda), y\xleftarrow{\$}\left\{C_{k'}^{r-1}\right\}^{-1}(0)} \approx_c \{\mathsf{iO}(C_{k'_g,y,\vec{1}}; t), x\}_{t\xleftarrow{\$}\{0,1\}^a, g\leftarrow\mathcal{U}_{\mathcal{F}^{r-1}}, y\xleftarrow{\$}g^{-1}(0)}.$$

Therefore to conclude Claim 74, it is enough to prove that

$$\{\mathsf{iO}(C_{k_f}^r; t), x\}_{t\xleftarrow{\$}\{0,1\}^a, f\leftarrow\mathcal{U}_{\mathcal{F}^r}, x\xleftarrow{\$}f^{-1}(1)} \approx_c \{\mathsf{iO}(C_{k'_g,y,\vec{1}}; t), x\}_{t\xleftarrow{\$}\{0,1\}^a, g\leftarrow\mathcal{U}_{\mathcal{F}^{r-1}}, y\xleftarrow{\$}g^{-1}(0)}.$$

This is the same as proving the following claim:

Claim 75.

$$\{\mathsf{iO}(C_{k_f}^r), x\}_{(f,x)\xleftarrow{\$}\mathrm{F}_\lambda^{0,r}} \approx_c \{\mathsf{iO}(C_{k'_g,y,\vec{1}}), y\}_{(g,y)\xleftarrow{\$}\mathrm{F}_\lambda^{1,r-1}},$$

where $\mathrm{F}_\lambda^{v,b} = \{(f,z) \mid f\in\mathcal{F}_\lambda^v, f(z)=b\}$*, for every* $v\in\mathbb{N}$*,* $b\in\{0,1\}$*,* $s\in\mathcal{K}_\lambda^t$*.*

Proof of Claim 75 Note that for every fixed pair $(f^*, x^*)\in\mathrm{F}_\lambda^{r,b}$, there exists a unique $(\tilde{g},\tilde{y})\in\mathrm{F}_\lambda^{r-1,0}$, and vice versa, such that $C_{k'_{\tilde{g}},\tilde{y},\vec{1}}$ has the same functionality as $C_{k_f}^r$ and $\tilde{y}=x^*$. In other words, there is a bijection $\mathcal{B}:\mathrm{F}_\lambda^{r,1}\mapsto\mathrm{F}_\lambda^{r-1,0}$ mapping (f^*,x^*) to $(\tilde{g},\tilde{y})$ such that $C_{k'_{\tilde{g}},\tilde{y},\vec{1}}$ has the same functionality as $C_{f^*}^r$ and $\tilde{y}=x^*$. In particular, $\tilde{y}=x^*$ and $\tilde{g}$ is the unique function that satisfies $\tilde{g}(x^*)=1$ and $\tilde{g}(x)=f^*(x)$ for every $x\neq x^*$.

By iO guarantees, this implies that for every fixed pair $(f^*,x^*)\in\mathrm{F}_\lambda^{r,b}$, the image under the bijection $\mathcal{B}$, $(\tilde{g},\tilde{y})\in\mathrm{F}_\lambda^{r-1,0}$, satisfies

$$\mathsf{iO}(C_{k_{f^*}}^r), x^* \approx_c \mathsf{iO}(C_{k'_{\tilde{g}},\tilde{y},\vec{1}}), y.$$

Therefore,

$$\{\mathsf{iO}(C_{k_f}^r), x\}_{(k,x)\xleftarrow{\$}\mathrm{F}_\lambda^{r,1}} \approx_c \{\mathsf{iO}(C_{k'_g,y,\vec{1}}), y\}_{(k',y)=\mathcal{B}(h,z),(h,z)\xleftarrow{\$}\mathrm{F}_\lambda^{0,k}} = \{\mathsf{iO}(C_{k'_g,y,\vec{1}}), y\}_{(k',y)\xleftarrow{\$}\mathrm{F}_\lambda^{r-1,0}},$$

where the last equality holds because $\mathcal{B}$ is a bijection. □

Corollary 76. *In particular, point functions form a* preimage-samplable *evasive function class with respect to the uniform distribution, i.e,* $(\mathcal{U}_{\mathcal{F}^1},\mathcal{F}^1)$ *is* preimage-samplable *evasive.*

Theorem 77. *Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ equipped with a distribution $\mathcal{D}_\mathcal{F}$ be a* preimage-samplable *evasive function class (see Definition 72) with input-length $n = n(\lambda)$, and $(\mathcal{D}, \mathfrak{C}^\mathcal{F})$ as the corresponding keyed circuit implementation for the* preimage-samplable *condition (see Definition 72).*

Assuming a $\mathsf{Id}_\mathcal{U}$-generalized unclonable puncturable obfuscation UPO *for any generalized puncturable keyed circuit class in* P/poly *(see Section 3.1.1), there is a copy-protection scheme for $\mathcal{F}$ that satisfies $(\mathcal{D}_\mathcal{F}, \mathcal{D}_{\mathsf{identical}})$-anti-piracy (see Appendix A.1) with respect to $\mathfrak{C}^\mathcal{F}$ as the keyed circuit implementation of $\mathcal{F}$, and $(\mathcal{D}, \mathfrak{C}^\mathcal{F})$ as the keyed circuit implementation of $(\mathcal{D}_\mathcal{F}, \mathcal{F})$, where* CopyProtect() *is the same as* UPO.Obf()*, and the distribution $\mathcal{D}_{\mathsf{identical}}$ on pairs of inputs is as follows:*

- *With probability $\frac{1}{2}$, output $(x_0^\mathcal{B}, x_0^\mathcal{C}) = (x, x)$, where $x \xleftarrow{\$} \{0,1\}^n$.*
- *With probability $\frac{1}{2}$, output $(x_1^\mathcal{B}, x_1^\mathcal{C}) = (x, x)$, where $x \xleftarrow{\$} {C_k^\mathcal{F}}^{-1}(1)$, and $C_k^\mathcal{F} \in \mathfrak{C}^\mathcal{F}$ is the circuit that is copy-protected.*

Proof of Theorem 77. The correctness of the copy-protection scheme follows directly from the correctness of the UPO.

We fix the keyed circuit representation of $(\mathcal{D}_\mathcal{F}, \mathcal{F})$ to be $(\mathcal{D}, \mathfrak{C}^\mathcal{F})$. Let the keyspace of $\mathfrak{C}^\mathcal{F}$ be $\mathcal{K}^\mathcal{F}$, i.e., $\mathfrak{C}^\mathcal{F} = \{\{{C^\mathcal{F}}_k\}_{k \in {\mathcal{K}^\mathcal{F}}_\lambda}\}_{\lambda \in \mathbb{N}}$.

Let $\mathfrak{C} = \{\{C_k\}_{k \in \mathcal{K}_\lambda}\}_{\lambda \in \mathbb{N}}$ be the auxiliary generalized puncturable keyed circuit class and $\mathcal{D}'$ be the corresponding distribution on $\mathcal{K}$ with respect to which the preimage-samplable condition (see Definition 72) holds for $(\mathcal{D}_\mathcal{F}, \mathcal{F})$ equipped with the keyed circuit description $(\mathcal{D}, \mathfrak{C}^\mathcal{F})$. Let Evasive-GenPuncture be the generalized puncturing algorithm associated with $\mathfrak{C}$.

We give a reduction from the copy-protection security experiment to the generalized unclonable puncturable obfuscation security experiment of UPO for the generalized puncturable keyed circuit class $\mathfrak{C}$ (see Figure 3). Let $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ be an adversary in the copy-protection security experiment. We mark the changes in blue.

<u>Hybrid$_0$</u>:
This is the same as the original copy-protection security experiment for the scheme (Obf, Eval).

- Ch samples a bit $b \xleftarrow{\$} \{0,1\}$.
- Ch samples $k \leftarrow \mathcal{D}(1^\lambda)$ $\rho_k \leftarrow \mathsf{UPO.Obf}(1^\lambda, {C^\mathcal{F}}_k)$ and sends it to $\mathcal{A}$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- Ch samples $x_0 \xleftarrow{\$} \{0,1\}^n$ and $x_1 \xleftarrow{\$} {C_k^\mathcal{F}}^{-1}(1)$.
- Apply $(\mathcal{B}(x_b, \cdot) \otimes \mathcal{C}(x_b, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_\mathcal{B}, b_\mathcal{C})$.
- Output 1 if $C_k^\mathcal{F}(x_b) = b_\mathcal{B} = b_\mathcal{C}$.

<u>Hybrid$_1$</u>:

- Ch samples a bit $b \xleftarrow{\$} \{0,1\}$.
- Ch samples $k \leftarrow \mathcal{D}(1^\lambda)$ $\rho_k \leftarrow \mathsf{UPO.Obf}(1^\lambda, {C^\mathcal{F}}_k)$ and sends it to $\mathcal{A}$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.

- Ch samples $x_0 \xleftarrow{\$} \{0,1\}^n$ and $x_1 \xleftarrow{\$} \{C_k^{\mathcal{F}}\}^{-1}(1)$.
- Apply $(\mathcal{B}(x_b,\cdot)\otimes\mathcal{C}(x_b,\cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_{\mathcal{B}}, b_{\mathcal{C}})$.
- Output 1 if ~~$C_k^{\mathcal{F}}(x_b) = b_{\mathcal{B}} = b_{\mathcal{C}}$~~ $b = b_{\mathcal{B}} = b_{\mathcal{C}}$.

Since $\mathcal{F}$ is evasive with respect to $\mathcal{D}$, with overwhelming probability $C_k^{\mathcal{F}}(x_0) = 0$. Hence, in the $b = 0$ case outputting 1 if $C_k^{\mathcal{F}}(x_0) = b_{\mathcal{B}} = b_{\mathcal{C}}$ is indistinguishable from $0 = b_{\mathcal{B}} = b_{\mathcal{C}}$. Clearly, since $x_1 \in C_k^{\mathcal{F}^{-1}}(1)$, in the $b = 1$ case, $C_k^{\mathcal{F}}(x_1) = b_{\mathcal{B}} = b_{\mathcal{C}}$ is the same as $1 = b_{\mathcal{B}} = b_{\mathcal{C}}$. Hence, the indistinguishability between Hybrid_0 and Hybrid_1 holds.

$\underline{\mathsf{Hybrid}_2}$:

- Ch samples a bit $b \xleftarrow{\$} \{0,1\}$.
- Ch samples ~~$k \leftarrow \mathcal{D}(1^\lambda)$~~ $k' \leftarrow \mathcal{D}'(1^\lambda), y \xleftarrow{\$} \{0,1\}^n$ and generates ~~$\rho_k \leftarrow \mathsf{UPO.Obf}(1^\lambda, C_k)$~~ $\rho_{k',y} \leftarrow \mathsf{UPO.Obf}(1^\lambda, C_{k',y})$, where $C_{k',y} \leftarrow \mathsf{Evasive\text{-}GenPuncture}(k', y, y, \vec{1}, \vec{1})$, and sends it to $\mathcal{A}$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- Ch samples $x_0 \xleftarrow{\$} \{0,1\}^n$ and ~~$x_1 \xleftarrow{\$} C_k^{\mathcal{F}^{-1}}(1)$~~ set $x_1 = y$.
- Apply $(\mathcal{B}(x_b,\cdot)\otimes\mathcal{C}(x_b,\cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_{\mathcal{B}}, b_{\mathcal{C}})$.
- Output 1 if $b = b_{\mathcal{B}} = b_{\mathcal{C}}$.

The indistinguishability between Hybrid_1 and Hybrid_2 holds by the $\mathsf{preimage\text{-}samplable}$ relation (in particular, Equation (8) for the $b = 1$ and $b = 0$ cases) between $\mathcal{F}, \mathcal{D}$ and $\mathcal{G}, \mathcal{D}'$.

$\underline{\mathsf{Hybrid}_3}$:

- Ch samples a bit $b \xleftarrow{\$} \{0,1\}$.
- Ch samples $k' \leftarrow \mathcal{D}'(1^\lambda), y \xleftarrow{\$} \{0,1\}^n$ and ~~generates $\rho_{k',y} \leftarrow \mathsf{UPO.Obf}(1^\lambda, C_{k',y})$, where $C_{k',y} \leftarrow \mathsf{Evasive\text{-}GenPuncture}(k', y, y, \vec{1}, \vec{1})$,~~ if $b = 0$ generates $\rho_{k'} \leftarrow \mathsf{Obf}(1^\lambda, C_{k'})$ else if $b = 1$ generates $\rho_{k',y} \leftarrow \mathsf{UPO.Obf}(1^\lambda, C_{k',y})$, where $C_{k',y} \leftarrow \mathsf{Evasive\text{-}GenPuncture}(k', y, y, \vec{1}, \vec{1})$, and sends it to $\mathcal{A}$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- Ch samples $x_0 \xleftarrow{\$} \{0,1\}^n$ and set $x_1 = y$.
- Apply $(\mathcal{B}(x_b,\cdot)\otimes\mathcal{C}(x_b,\cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_{\mathcal{B}}, b_{\mathcal{C}})$.
- Output 1 if $b = b_{\mathcal{B}} = b_{\mathcal{C}}$.

The indistinguishability between Hybrid_2 and Hybrid_3 holds as follows. In the $b = 0$ case of Hybrid_2, the view of $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ only depends on $\mathsf{UPO.Obf}(1^\lambda, C_{k',y}), x_0$, but in the $b = 0$ case of

Hybrid_3, the view depends on $\mathsf{UPO.Obf}(1^\lambda, C_{k'}), x_0$ where $x_0 \xleftarrow{\$} \{0,1\}^n$ is sampled independent of k' and y. Hence it is enough to show that

$$\{\mathsf{UPO.Obf}(1^\lambda, C_{k',y})\}_{k' \leftarrow \mathcal{D}'(1^\lambda), y \xleftarrow{\$} \{0,1\}^n} \approx_c \{\mathsf{UPO.Obf}(1^\lambda, C_{k'})\}_{k' \leftarrow \mathcal{D}'(1^\lambda)}, \tag{11}$$

which is a necessary condition for the generalized unclonable puncturable obfuscation security of UPO (otherwise $\mathcal{A}$ can itself distinguish between $b = 0$ and $b = 1$ case in the generalized unclonable puncturable obfuscation security experiment given in Definition 9 for the keyed circuitclass $\mathfrak{C}$). Therefore, Equation (11) holds by the generalized UPO security of UPO for the circuit class $\mathfrak{C}$.

<u>Hybrid_4</u>:

- Ch samples a bit $b \xleftarrow{\$} \{0,1\}$.
- Ch samples $k' \leftarrow \mathcal{D}'(1^\lambda), y \xleftarrow{\$} \{0,1\}^n$ and if $b = 0$ generates $\rho_{k'} \leftarrow \mathsf{Obf}(1^\lambda, C_{k'})$ else if $b = 1$ generates $\rho_{k',y} \leftarrow \mathsf{UPO.Obf}(1^\lambda, C_{k',y})$, where $C_{k',y} \leftarrow \mathsf{Evasive\text{-}GenPuncture}(k', y, y, \vec{1}, \vec{1})$, and sends it to $\mathcal{A}$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- ~~Ch samples $x_0 \xleftarrow{\$} \{0,1\}^n$ and set $x_1 = y$.~~
- Apply $(\mathcal{B}(\cancel{x_b}y, \cdot) \otimes \mathcal{C}(\cancel{x_b}y, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_\mathcal{B}, b_\mathcal{C})$.
- Output 1 if $b = b_\mathcal{B} = b_\mathcal{C}$.

The only change from Hybrid_3 to Hybrid_4 is replacing x_0 with y in the $b = 0$ case and x_1 with y in the $b = 1$ case. The indistinguishability between Hybrid_3 and Hybrid_4 holds as follows. Note that replacing x_1 with y in Hybrid_3 does not change anything since x_1 was set to y in Hybrid_3. Next, in the $b = 1$ case, the view of $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ only depends on $\mathsf{UPO.Obf}(1^\lambda, C_{k'}), x_0$ where $x_0 \xleftarrow{\$} \{0,1\}^n$ is sampled independent of k'. Since $y \xleftarrow{\$} \{0,1\}^n$ is also sampled independent of k',

$$\{C_{k'}, x_0\}_{k' \leftarrow \mathcal{D}'(1^\lambda), x_0 \xleftarrow{\$} \{0,1\}^n} = \{C_{k'}, y\}_{k' \leftarrow \mathcal{D}'(1^\lambda), y \xleftarrow{\$} \{0,1\}^n}.$$

Hence,

$$\begin{aligned} &\{\mathsf{UPO.Obf}(1^\lambda, C_{k'}), x_0\}_{k' \leftarrow \mathcal{D}'(1^\lambda), x_0 \xleftarrow{\$} \{0,1\}^n} \\ &= \{\mathsf{UPO.Obf}(1^\lambda, C_{k'}), y\}_{k' \leftarrow \mathcal{D}'(1^\lambda), y \xleftarrow{\$} \{0,1\}^n}. \end{aligned}$$

Therefore, replacing $\mathsf{UPO.Obf}(1^\lambda, C_{k'}), x_0$ with $\mathsf{UPO.Obf}(1^\lambda, C_{k'}), y$ is indistinguishable and hence, Hybrid_3 and Hybrid_4 are indistinguishable with respect to the adversary.

We next give a reduction $(\mathcal{R}_\mathcal{A}, \mathcal{R}_\mathcal{B}, \mathcal{R}_\mathcal{C})$ from Hybrid_4 to the $\mathsf{Id}_\mathcal{U}$-generalized UPO security experiment of UPO (Definition 9) for the generalized puncturable keyed circuitclass $\mathfrak{C} = \{\{C_{k'}\}_{k' \in \mathcal{K}_\lambda}\}_\lambda$ equipped with $\mathsf{Evasive\text{-}GenPuncture}$ as the generalized puncturing algorithm (see Appendix A.2).

- $\mathcal{R}_\mathcal{A}$ samples $k' \leftarrow \mathcal{D}'(1^\lambda)$, and sends k' along with $\mu_\mathcal{B} = \mu_\mathcal{C} = \vec{1}$, the constant 1 function.

- On receiving ρ from Ch, the challenger for the generalized unclonable puncturable obfuscation experiment, $\mathcal{R}_{\mathcal{A}}$ runs $\mathcal{A}(\rho)$ to get a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$, and sends $\sigma_{\mathcal{B}}, \sigma_{\mathcal{C}}$ to $\mathcal{R}_{\mathcal{B}}$ and $\mathcal{R}_{\mathcal{C}}$ respectively.

- $\mathcal{R}_{\mathcal{B}}$ (respectively, $\mathcal{R}_{\mathcal{C}}$) runs $\mathcal{B}(x_{\mathcal{B}}, \sigma_{\mathcal{B}})$ (respectively, $\mathcal{C}(x_{\mathcal{C}}, \sigma_{\mathcal{C}})$) on receiving $x_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}$ (respectively $x_{\mathcal{C}}$ and $\sigma_{\mathcal{C}}$) from Ch and $\mathcal{R}_{\mathcal{A}}$, respectively, and output the outcome.

Clearly, the view of $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ in the experiment (Figure 3) $\mathsf{GenUPO.Expt}^{(\mathcal{R}_{\mathcal{A}}, \mathcal{R}_{\mathcal{B}}, \mathcal{R}_{\mathcal{C}}), \mathsf{Id}_{\mathcal{U}}, \mathfrak{C}}(1^{\lambda}, 0)$ (respectively, $\mathsf{GenUPO.Expt}^{(\mathcal{R}_{\mathcal{A}}, \mathcal{R}_{\mathcal{B}}, \mathcal{R}_{\mathcal{C}}), \mathsf{Id}_{\mathcal{U}}, \mathfrak{C}}(1^{\lambda}, 1)$) is exactly the same as that in the $b = 0$ (respectively, $b = 1$) case in Hybrid_4, where $\mathsf{Id}_{\mathcal{U}}$ is as defined in Section 3.1.1. This completes the reduction from the copy-protection security experiment to the generalized unclonable puncturable obfuscation security experiment (Figure 3). □

Corollary 78. *Suppose r is such that the following holds:*

1. *$\mathcal{F}^r$ is evasive with respect to $\mathcal{U}_{\mathcal{F}^r}$, the uniform distribution.*

2. *There exists a keyed circuit implementation $(\mathcal{D}^r, \mathfrak{C}^r)$ for $(\mathcal{U}_{\mathcal{F}^r}, \mathcal{F}^r)$, and similarly keyed circuit implementation $(\mathcal{D}^{r-1}, \mathfrak{C}^{r-1})$ for $(\mathcal{U}_{\mathcal{F}^{r-1}}, \mathcal{F}^{r-1})$.*

Then, assuming a $\mathsf{Id}_{\mathcal{U}}$-generalized unclonable puncturable obfuscation UPO for any generalized puncturable keyed circuit class in $\mathsf{P/poly}$ (see Section 3.1.1), there is a copy-protection scheme for $\mathcal{F}^r$ that satisfies $(\mathcal{U}_{\mathcal{F}^r}, \mathcal{D}_{\mathsf{identical}})$-anti-piracy (see Appendix A.1) with respect to some keyed circuit implementation $(\mathcal{D}, \mathfrak{C})$ of $(\mathcal{U}_{\mathcal{F}^r}, \mathcal{F})$, where $\mathsf{CopyProtect}()$ is the same as $\mathsf{UPO.Obf}()$, and the distribution $\mathcal{D}_{\mathsf{identical}}$ on pairs of inputs is as follows:

- *With probability $\frac{1}{2}$, output $(x_0^{\mathcal{B}}, x_0^{\mathcal{C}}) = (x, x)$, where $x \xleftarrow{\$} \{0,1\}^n$.*

- *With probability $\frac{1}{2}$, output $(x_1^{\mathcal{B}}, x_1^{\mathcal{C}}) = (x, x)$, where $x \xleftarrow{\$} {C_k}^{-1}(1)$, and $C_k \in \mathfrak{C}$ is the circuit that is copy-protected.*

Then, assuming post-quantum indistinguishability obfuscation, and a $\mathsf{Id}_{\mathcal{U}}$-generalized unclonable puncturable obfuscation UPO for any generalized puncturable keyed circuit class in $\mathsf{P/poly}$ (see Section 3.1.1), there exists a copy-protection for $\mathcal{F}^r$ that satisfies $(\mathcal{U}, \mathcal{D}_{\mathsf{identical}})$-anti-piracy (see Appendix A.1), where $\mathsf{CopyProtect}() = \mathsf{UPO.Obf}()$, and the distribution $\mathcal{D}_{\mathsf{identical}}$ on pairs of inputs is as follows:

- *With probability $\frac{1}{2}$, output $(x_0^{\mathcal{B}}, x_0^{\mathcal{C}}) = (x, x)$, where $x \xleftarrow{\$} \{0,1\}^n$.*

- *With probability $\frac{1}{2}$, output $(x_1^{\mathcal{B}}, x_1^{\mathcal{C}}) = (x, x)$, where $x \xleftarrow{\$} {C_k^{\mathcal{F}}}^{-1}(1)$, and $C_k^{\mathcal{F}} \in \mathcal{F}$ is the circuit that is copy-protected.*

In particular, there exists a copy-protection for point functions that satisfies $(\mathcal{U}, \mathcal{D}_{\mathsf{identical}})$-anti-piracy, under the assumptions made above.

Acknowledgements

We thank Supartha Podder for discussions during the early stages of the project.

References

[Aar09] Scott Aaronson. "Quantum copy-protection and quantum money". In: *2009 24th Annual IEEE Conference on Computational Complexity*. IEEE. 2009, pp. 229–242 (cit. on pp. 3, 99).

[Aar16] Scott Aaronson. *The Complexity of Quantum States and Transformations: From Quantum Money to Black Holes*. 2016. arXiv: 1607.05256 [quant-ph] (cit. on pp. 15, 38).

[AC12] Scott Aaronson and Paul Christiano. "Quantum Money from Hidden Subspaces". In: *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*. STOC '12. New York, New York, USA: Association for Computing Machinery, 2012, pp. 41–60. ISBN: 9781450312455. DOI: 10.1145/2213977.2213983. URL: https://doi.org/10.1145/2213977.2213983 (cit. on pp. 3, 99).

[AK21] Prabhanjan Ananth and Fatih Kaleoglu. "Unclonable Encryption, Revisited". In: *Theory of Cryptography Conference*. Springer. 2021, pp. 299–329 (cit. on pp. 7, 82, 100).

[AK22] Prabhanjan Ananth and Fatih Kaleoglu. "A note on copy-protection from random oracles". In: *arXiv preprint arXiv:2208.12884* (2022) (cit. on p. 99).

[AKL+22] Prabhanjan Ananth, Fatih Kaleoglu, Xingjian Li, Qipeng Liu, and Mark Zhandry. "On the feasibility of unclonable encryption, and more". In: *Annual International Cryptology Conference*. Springer. 2022, pp. 212–241 (cit. on pp. 3, 7, 100).

[AKL23] Prabhanjan Ananth, Fatih Kaleoglu, and Qipeng Liu. "Cloning Games: A General Framework for Unclonable Primitives". In: *arXiv preprint arXiv:2302.01874* (2023) (cit. on pp. 5, 7, 8, 20, 29, 52, 94, 100).

[AL21] Prabhanjan Ananth and Rolando L. La Placa. "Secure Software Leasing". In: *Advances in Cryptology – EUROCRYPT 2021*. Ed. by Anne Canteaut and François-Xavier Standaert. Cham: Springer International Publishing, 2021, pp. 501–530. ISBN: 978-3-030-77886-6 (cit. on pp. 3, 7, 99).

[ALL+21] Scott Aaronson, Jiahui Liu, Qipeng Liu, Mark Zhandry, and Ruizhe Zhang. "New Approaches for Quantum Copy-Protection". In: *Advances in Cryptology – CRYPTO 2021*. Ed. by Tal Malkin and Chris Peikert. Cham: Springer International Publishing, 2021, pp. 526–555. ISBN: 978-3-030-84242-0 (cit. on pp. 3, 99).

[BGI+01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. "On the (im) possibility of obfuscating programs". In: *Annual international cryptology conference*. Springer. 2001, pp. 1–18 (cit. on pp. 3, 13, 18, 101).

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. "Functional signatures and pseudorandom functions". In: *International workshop on public key cryptography*. Springer. 2014, pp. 501–519 (cit. on pp. 6, 55).

[BGS13] Anne Broadbent, Gus Gutoski, and Douglas Stebila. "Quantum one-time programs". In: *Annual Cryptology Conference*. Springer. 2013, pp. 344–360 (cit. on p. 3).

[BI20] Anne Broadbent and Rabib Islam. "Quantum Encryption with Certified Deletion". In: *Theory of Cryptography*. Springer International Publishing, 2020, pp. 92–122. DOI: 10.1007/978-3-030-64381-2_4. URL: https://doi.org/10.1007%2F978-3-030-64381-2_4 (cit. on p. 99).

[BKL23] Anne Broadbent, Martti Karvonen, and Sébastien Lord. "Uncloneable Quantum Advice". In: *arXiv preprint arXiv:2309.05155* (2023) (cit. on p. 3).

[BL20] Anne Broadbent and Sébastien Lord. "Uncloneable Quantum Encryption via Oracles". en. In: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. DOI: 10.4230/LIPICS.TQC.2020.4. URL: https://drops.dagstuhl.de/opus/volltexte/2020/12063/ (cit. on pp. 3, 7, 99, 100).

[BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. "On the cryptographic hardness of finding a Nash equilibrium". In: *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE. 2015, pp. 1480–1498 (cit. on p. 3).

[BS16] Shalev Ben-David and Or Sattath. *Quantum Tokens for Digital Signatures*. 2016. DOI: 10.48550/ARXIV.1609.09047. URL: https://arxiv.org/abs/1609.09047 (cit. on p. 3).

[BS20] Amit Behera and Or Sattath. "Almost public quantum coins". In: *arXiv preprint arXiv:2002.12438* (2020) (cit. on p. 3).

[BSW16] Mihir Bellare, Igors Stepanovs, and Brent Waters. "New Negative Results on Differing-Inputs Obfuscation". In: *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. Lecture Notes in Computer Science. Springer, 2016, pp. 792–821. DOI: 10.1007/978-3-662-49896-5_28. URL: https://doi.org/10.1007/978-3-662-49896-5%5C_28 (cit. on p. 58).

[BW13] Dan Boneh and Brent Waters. "Constrained pseudorandom functions and their applications". In: *Advances in Cryptology-ASIACRYPT 2013: 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II 19*. Springer. 2013, pp. 280–300 (cit. on pp. 6, 55).

[BZ17] Dan Boneh and Mark Zhandry. "Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation". In: *Algorithmica* 79 (2017), pp. 1233–1285 (cit. on p. 3).

[CG23] Andrea Coladangelo and Sam Gunn. "How to Use Quantum Indistinguishability Obfuscation". In: *arXiv preprint arXiv:2311.07794* (2023) (cit. on p. 8).

[CHV23] Céline Chevalier, Paul Hermouet, and Quoc-Huy Vu. "Semi-Quantum Copy-Protection and More". In: *Cryptology ePrint Archive* (2023) (cit. on pp. 6, 7, 99).

[CLLZ21] Andrea Coladangelo, Jiahui Liu, Qipeng Liu, and Mark Zhandry. "Hidden Cosets and Applications to Unclonable Cryptography". In: *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part I*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. Lecture Notes in Computer Science. Springer, 2021, pp. 556–584. DOI:

10.1007/978-3-030-84242-0_20. URL: https://doi.org/10.1007/978-3-030-84242-0%5C_20 (cit. on pp. 3, 6, 7, 12, 13, 21, 22, 25, 29, 30, 32, 33, 35–39, 50, 95, 99, 100).

[Die82] DGBJ Dieks. "Communication by EPR devices". In: *Physics Letters A* 92.6 (1982), pp. 271–272 (cit. on p. 3).

[Gao15] Jingliang Gao. "Quantum union bounds for sequential projective measurements". In: *Physical Review A* 92.5 (2015), p. 052331 (cit. on p. 15).

[GGH+16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. "Candidate indistinguishability obfuscation and functional encryption for all circuits". In: *SIAM Journal on Computing* 45.3 (2016), pp. 882–929 (cit. on p. 3).

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. "Two-round secure MPC from indistinguishability obfuscation". In: *Theory of Cryptography Conference*. Springer. 2014, pp. 74–94 (cit. on p. 3).

[GMR23] Vipul Goyal, Giulio Malavolta, and Justin Raizes. "Unclonable Commitments and Proofs". In: *Cryptology ePrint Archive* (2023) (cit. on p. 3).

[Got02] Daniel Gottesman. "Uncloneable Encryption". In: (2002). DOI: 10.48550/ARXIV.QUANT-PH/0210062. URL: https://arxiv.org/abs/quant-ph/0210062 (cit. on p. 3).

[GZ20] Marios Georgiou and Mark Zhandry. *Unclonable Decryption Keys*. 2020. IACR Cryptol. ePrint Arch. https://eprint.iacr.org/2020/877 (cit. on pp. 3, 5, 7, 21, 82, 97, 99, 100).

[JK23] Ruta Jawale and Dakshita Khurana. "Unclonable Non-Interactive Zero-Knowledge". In: *arXiv preprint arXiv:2310.07118* (2023) (cit. on p. 3).

[KN23] Fuyuki Kitagawa and Ryo Nishimaki. "One-out-of-Many Unclonable Cryptography: Definitions, Constructions, and More". In: *arXiv preprint arXiv:2302.09836* (2023) (cit. on pp. 3, 100).

[KT22] Srijita Kundu and Ernest Y-Z Tan. "Device-independent uncloneable encryption". In: *arXiv preprint arXiv:2210.01058* (2022) (cit. on pp. 8, 20).

[LLQZ22] Jiahui Liu, Qipeng Liu, Luowen Qian, and Mark Zhandry. "Collusion Resistant Copy-Protection for Watermarkable Functionalities". In: *Theory of Cryptography - 20th International Conference, TCC 2022, Chicago, IL, USA, November 7-10, 2022, Proceedings, Part I*. Ed. by Eike Kiltz and Vinod Vaikuntanathan. Vol. 13747. Lecture Notes in Computer Science. Springer, 2022, pp. 294–323. DOI: 10.1007/978-3-031-22318-1_11. URL: https://doi.org/10.1007/978-3-031-22318-1%5C_11 (cit. on pp. 6, 59, 99).

[LMZ23] Jiahui Liu, Hart Montgomery, and Mark Zhandry. "Another Round of Breaking and Making Quantum Money: How to Not Build It from Lattices, and More". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2023, pp. 611–638 (cit. on pp. 3, 7, 99).

[NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667 (cit. on p. 13).

[RS19] Roy Radian and Or Sattath. "Semi-quantum money". In: *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*. 2019, pp. 132–146 (cit. on p. 3).

[RZ21] Bhaskar Roberts and Mark Zhandry. "Franchised quantum money". In: *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part I 27*. Springer. 2021, pp. 549–574 (cit. on p. 3).

[Shm22] Omri Shmueli. "Public-key Quantum money with a classical bank". In: *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*. 2022, pp. 790–803 (cit. on pp. 3, 99).

[SW14] Amit Sahai and Brent Waters. "How to use indistinguishability obfuscation: deniable encryption, and more". In: *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*. 2014, pp. 475–484 (cit. on pp. 3, 6, 12, 61, 62, 82).

[SW22] Or Sattath and Shai Wyborski. *Uncloneable Decryptors from Quantum Copy-Protection*. 2022. arXiv: 2203.05866 (cit. on p. 97).

[Wie83] Stephen Wiesner. "Conjugate coding". In: *ACM Sigact News* 15.1 (1983), pp. 78–88 (cit. on pp. 3, 98).

[WZ82] William K Wootters and Wojciech H Zurek. "A single quantum cannot be cloned". In: *Nature* 299.5886 (1982), pp. 802–803 (cit. on p. 3).

[Zha19] Mark Zhandry. "Quantum Lightning Never Strikes the Same State Twice". In: *Advances in Cryptology – EUROCRYPT 2019*. Ed. by Yuval Ishai and Vincent Rijmen. Cham: Springer International Publishing, 2019, pp. 408–438. ISBN: 978-3-030-17659-4 (cit. on pp. 3, 7, 99).

[Zha23] Mark Zhandry. "Quantum Money from Abelian Group Actions". In: *arXiv preprint arXiv:2307.12120* (2023) (cit. on pp. 7, 99).

A Unclonable Cryptography: Definitions

A.1 Quantum Copy-Protection

Consider a function class $\mathcal{F}$ with keyed circuit implementation $\mathfrak{C} = \{\mathfrak{C}_\lambda\}_{\lambda\in\mathbb{N}}$, where $\mathcal{F}_\lambda$ (respectively, $\mathfrak{C}_\lambda$) consists of functions (respectively, circuits) with input length $n(\lambda)$ and output length $m(\lambda)$. A copy-protection scheme is a pair of QPT algorithms $(\mathsf{CopyProtect}, \mathsf{Eval})$ defined as follows:

- $\mathsf{CopyProtect}(1^\lambda, C)$: on input a security parameter λ and a circuit $C \in \mathfrak{C}_\lambda$, it outputs a quantum state ρ_C.
- $\mathsf{Eval}(\rho_k, x)$: on input a quantum state ρ_C and an input $x \in \mathcal{X}_\lambda$, it outputs (ρ'_C, y).

Correctness. A copy-protection scheme $(\mathsf{CopyProtect}, \mathsf{Eval})$ for a function class $\mathcal{F}$ with keyed circuit implementation $\mathfrak{C} = \{\mathfrak{C}_\lambda\}_{\lambda\in\mathbb{N}}$ is δ-correct, if for every $C \in \mathfrak{C}_\lambda$, for every $x \in \{0,1\}^{n(\lambda)}$, there exists a negligible function $\delta(\lambda)$ such that:

$$\Pr\left[C(x) = y \mid {}^{\rho_C \leftarrow \mathsf{CopyProtect}(1^\lambda, C)}_{(\rho'_C, y) \leftarrow \mathsf{Eval}(\rho_C, x)}\right] \geq 1 - \delta(\lambda)$$

$\underline{\mathsf{CP.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}_\mathcal{K},\mathcal{D}_\mathcal{X}}(1^\lambda)}$:

- Ch samples $k \leftarrow \mathcal{D}_\mathcal{K}(1^\lambda)$ and generates $\rho_k \leftarrow \mathsf{CopyProtect}(1^\lambda, C_k)$ and sends ρ_k to $\mathcal{A}$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- Ch samples $(x^\mathcal{B}, x^\mathcal{C}) \leftarrow \mathcal{D}_\mathcal{X}$[10].
- Apply $(\mathcal{B}(x^\mathcal{B}, \cdot) \otimes \mathcal{C}(x^\mathcal{C}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(y^\mathcal{B}, y^\mathcal{C})$.
- Output 1 if $y^\mathcal{B} = C(x^\mathcal{B})$ and $y^\mathcal{C} = C_k(x^\mathcal{C})$, else 0.

Figure 32: $(\mathcal{D}_\mathcal{K}, \mathcal{D}_\mathcal{X})$-anti-piracy experiment of copy-protection.

$(\mathcal{D}_\mathcal{K}, \mathcal{D}_\mathcal{X})$-anti-piracy. Consider the experiment in Figure 32. We define $p_{\mathsf{triv}} = \max\{p_\mathsf{B}, p_\mathsf{C}\}$, where p_B is the maximum probability that the experiment outputs 1 when $\mathcal{A}$ gives ρ_C to $\mathcal{B}$ and $\mathcal{C}$ outputs its best guess and $p_\mathcal{C}$ is defined symmetrically. We refer to [AKL23] for a formal definition of trivial success probability.

Suppose $\mathcal{D}_\mathcal{X}$ is a distribution on $\{0,1\}^{n(\lambda)} \times \{0,1\}^{n(\lambda)}$, and $\mathcal{D}_\mathcal{F}$ is a distribution on $\mathcal{F}$.

We say that a copy-protection scheme $(\mathsf{CopyProtect}, \mathsf{Eval})$ for $\mathcal{F}$ satisfies $(\mathcal{D}_\mathcal{F}, \mathcal{D}_\mathcal{X})$-anti-piracy if there exists a keyed circuit implementation (see Section 6.1) of the form $(\mathcal{D}_\mathcal{K}, \mathfrak{C})$[11] for $(\mathcal{D}_\mathcal{F}, \mathcal{F})$

[10] $\mathcal{D}_\mathcal{X}$ may potentially depend on the circuit C_k.

[11] It is crucial that $\mathfrak{C}$ is the same circuit class as the keyed implementation of $\mathcal{F}$ that we fixed

such that for every tuple of QPT adversaries $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that:

$$\Pr\left[1 \leftarrow \mathsf{CP.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}_{\mathcal{K}},\mathcal{D}_{\mathcal{X}}}\left(1^{\lambda}\right)\right] \leq p_{\mathsf{triv}} + \mathsf{negl}(\lambda)$$

If $\mathcal{D}_{\mathcal{X}}$ is a uniform distribution on $\{0,1\}^{n(\lambda)} \times \{0,1\}^{n(\lambda)}$ then we simply refer to this definition as $\mathcal{D}_{\mathcal{K}}$-anti-piracy.

A.2 Public-Key Single-Decryptor Encryption

We adopt the following definition of public-key single-decryptor encryption from [CLLZ21].

A public-key single-decryptor encryption scheme with message length $n(\lambda$ and ciphertext length $c(\lambda)$ consists of the QPT algorithms $\mathsf{SDE} = (\mathsf{Gen}, \mathsf{QKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ defined below:

- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^{\lambda})$: on input a security parameter 1^{λ}, returns a classical secret key sk and a classical public key pk.
- $\rho_{\mathsf{sk}} \leftarrow \mathsf{QKeyGen}(\mathsf{sk})$: takes a classical secret key sk and outputs a quantum decryptor key ρ_{sk}.
- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$ takes a classical public key pk, a message $m \in \{0,1\}^n$ and outputs a classical ciphertext ct.
- $m \leftarrow \mathsf{Dec}(\rho_{\mathsf{sk}}, \mathsf{ct})$: takes a quantum decryptor key ρ_{sk} and a ciphertext ct, and outputs a message $m \in \{0,1\}^n$.

Correctness For every message $m \in \{0,1\}^{n(\lambda)}$, there exists a negligible function $\delta(\lambda)$ such that:

$$\Pr\left[\mathsf{Dec}(\rho_{\mathsf{sk}}, \mathsf{ct}) = m \;\middle|\; \substack{(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{Gen}(\lambda) \\ \rho_{\mathsf{sk}} \leftarrow \mathsf{QKeyGen}(\mathsf{sk}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m)}\right] \geq 1 - \delta(\lambda).$$

$\underline{\mathsf{Search.SDE.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}}\left(1^{\lambda}\right)}$:

- Ch samples $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^{\lambda})$. It then generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{QKeyGen}(\mathsf{sk})$ and sends $(\rho_{\mathsf{sk}}, \mathsf{pk})$ to $\mathcal{A}$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- Ch samples $(m^{\mathcal{B}}, m^{\mathcal{C}}) \leftarrow \mathcal{D}(1^{\lambda})$ and generates $\mathsf{ct}^{\mathcal{B}} \leftarrow \mathsf{Enc}(\mathsf{pk}, m^{\mathcal{B}})$ and $\mathsf{ct}^{\mathcal{C}} \leftarrow \mathsf{Enc}(\mathsf{pk}, m^{\mathcal{C}})$.
- Apply $(\mathcal{B}(\mathsf{ct}^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(y^{\mathcal{B}}, y^{\mathcal{C}})$.
- Output 1 if $y^{\mathcal{B}} = m^{\mathcal{B}}$ and $y^{\mathcal{C}} = m^{\mathcal{C}}$.

Figure 33: Search anti-piracy.

Search Anti-Piracy We say that a single-decryptor encryption scheme SDE satisfies $\mathcal{D}$-search anti-piracy if for every QPT adversary $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ in Figure 33 if there exists a negligible function negl such that:

$$\Pr\left[1 \leftarrow \mathsf{Search.SDE.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C})}\left(1^\lambda\right)\right] \leq \mathsf{negl}(\lambda).$$

The two instantiations of $\mathcal{D}$ are $\mathcal{U}$ and $\mathsf{Id}_{\mathcal{U}}$, as defined in section 3.1.1.

$\underline{\mathsf{Ind\text{-}random.SDE.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}_{\mathsf{ct}}}\left(1^\lambda\right)}$:

- Ch samples $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda)$. It then generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{QKeyGen}(\mathsf{sk})$ and sends $(\rho_{\mathsf{sk}}, \mathsf{pk})$ to $\mathcal{A}$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- Ch samples $b \xleftarrow{\$} \{0,1\}$, and generates $(\mathsf{ct}_b^{\mathcal{B}}, \mathsf{ct}_b^{\mathcal{C}}) \leftarrow \mathcal{D}_{\mathsf{ct}}(1^\lambda, b, \mathsf{pk})$.
- Apply $(\mathcal{B}(\mathsf{ct}_b^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}_b^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.
- Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$.

Figure 34: Indistinguishability from random anti-piracy.

Indistinguishability from random Anti-Piracy We say that a single-decryptor encryption scheme SDE satisfies $\mathcal{D}_{\mathsf{ct}}$-indistinguishability from random anti-piracy if for every QPT adversary $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ in Figure 33 if there exists a negligible function negl such that:

$$\Pr\left[1 \leftarrow \mathsf{Ind\text{-}random.SDE.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}_{\mathsf{ct}}}\left(1^\lambda\right)\right] \leq \mathsf{negl}(\lambda).$$

The two instantiations of $\mathcal{D}_{\mathsf{ct}}$ are as follows:

1. $\mathcal{D}_{\mathsf{ind\text{-}msg}}(1^\lambda, b, \mathsf{pk})$:
   (a) Sample $m^{\mathcal{B}}, m^{\mathcal{C}} \xleftarrow{\$} \{0,1\}^q$, where $q(\lambda)$ is the message length.
   (b) Generate $\mathsf{ct}_b^{\mathcal{B}} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b^{\mathcal{B}})$ and $\mathsf{ct}_b^{\mathcal{C}} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b^{\mathcal{C}})$, where $m_0^{\mathcal{B}} = m_0^{\mathcal{C}} = 0$, $m_1^{\mathcal{B}} = m^{\mathcal{B}}$ and $m_1^{\mathcal{C}} = m^{\mathcal{C}}$.
   (c) Output $\mathsf{ct}_b^{\mathcal{B}}, \mathsf{ct}_b^{\mathcal{C}}$.
2. $\mathcal{D}_{\mathsf{identical\text{-}cipher}}(1^\lambda, b, \mathsf{pk})$:
   (a) Sample $m \xleftarrow{\$} \{0,1\}^q$, where $q(\lambda)$ is the message length.
   (b) Generate $\mathsf{ct}_b \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$ where $m_0 = 0$, and $m_1 = m$.
   (c) Set $\mathsf{ct}_b^{\mathcal{B}} = \mathsf{ct}_b^{\mathcal{C}} = \mathsf{ct}_b$.
   (d) Output $\mathsf{ct}_b^{\mathcal{B}}, \mathsf{ct}_b^{\mathcal{C}}$.

$\underline{\mathsf{SelCPA.SDE.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}}\left(1^\lambda\right)}$:

1. Ch samples $(\rho_k, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and sends ρ_k to $\mathcal{A}$.
2. $\mathcal{A}(\rho_k)$ outputs $(m_0^{\mathcal{B}}, m_1^{\mathcal{B}}, m_0^{\mathcal{C}}, m_1^{\mathcal{C}})$, such that $|m_0^{\mathcal{B}}| = |m_1^{\mathcal{B}}|$ and $|m_0^{\mathcal{C}}| = |m_1^{\mathcal{C}}|$, to challenger and then bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
3. Ch samples $b \xleftarrow{\$} \{0,1\}$.
4. Let $\mathsf{ct}^{\mathcal{B}}, \mathsf{ct}^{\mathcal{C}} \leftarrow \mathcal{D}(1^\lambda, b, \mathsf{pk})$.
5. Apply $(\mathcal{B}(\mathsf{ct}^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b_{\mathbf{B}}, b_{\mathbf{C}})$.
6. Output 1 if $b_{\mathbf{B}} = b_{\mathbf{C}} = b$.

Figure 35: Selective $\mathcal{D}$-CPA anti-piracy.

Selective CPA Anti-piracy We say that a single-decryptor encryption scheme SDE satisfies $\mathcal{D}$-selective CPA anti-piracy, for a distribution $\mathcal{D}$ on $\{0,1\}^n \times \{0,1\}^n$, if for every QPT adversary $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ in Figure 35, there exists a negligible function negl such that:

$$\Pr\left[1 \leftarrow \mathsf{SelCPA.SDE.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}}\left(1^\lambda\right)\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

The two instantiations of $\mathcal{D}$ are:

1. $\mathcal{D}_{\text{iden-bit,ind-msg}}(1^\lambda, b, \mathsf{pk})$: outputs $(\mathsf{ct}^{\mathcal{B}}, \mathsf{ct}^{\mathcal{C}})$ where $\mathsf{ct}^{\mathcal{B}} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b^{\mathcal{B}})$ and $\mathsf{ct}^{\mathcal{C}} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b^{\mathcal{C}})$.
2. $\mathcal{D}_{\text{identical}}(1^\lambda, b, \mathsf{pk})$ outputs $(\mathsf{ct}, \mathsf{ct})$ where $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b^{\mathcal{B}})$[12].

This notion of selective $\mathcal{D}_{\text{identical}}$-$\mathsf{CPA}$ security is equivalent to the selective CPA-security in [GZ20].

CPA anti-piracy We say that a single-decryptor encryption scheme SDE satisfies CPA $\mathcal{D}$-anti-piracy if for every QPT adversary $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ in Experiment 36, there exists a negligible function negl such that

$$\Pr\left[1 \leftarrow \mathsf{CPA.SDE.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}}\left(1^\lambda\right)\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

The two instantiations of $\mathcal{D}$ are $\mathcal{D}_{\text{iden-bit,ind-msg}}$ and $\mathcal{D}_{\text{identical}}$, defined in the selective CPA anti-piracy definition in the previous paragraph.
The definition of $\mathcal{D}_{\text{iden-bit,ind-msg}}$-$\mathsf{CPA}$ anti-piracy is the same as the correlated version of the 1-2 variant of $\mathsf{UD}-\mathsf{CPA}$ anti-piracy defined in [SW22] and the definition $\mathsf{Id}_{\mathcal{U}}$-$\mathsf{CPA}$ anti-piracy is the same as the secret-key CPA secure defined in [GZ20].

[12]Ideally, in the identical challenge setting, there should be just two challenge messages m_0, m_1 and not $m_0^{\mathcal{B}}, m_1^{\mathcal{B}}, m_0^{\mathcal{C}}, m_1^{\mathcal{C}}$, but we chose to have this redundancy in order to unify the syntax for the identical and correlated challenge settings.

$\underline{\mathsf{CPA.SDE.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C}),\mathcal{D}}\left(1^{\lambda}\right)}$:

- Ch samples $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^{\lambda})$ and generates $\rho_{\mathsf{sk}} \leftarrow \mathsf{QKeyGen}(\mathsf{sk})$ and sends $(\rho_{\mathsf{sk}}, \mathsf{pk})$ to $\mathcal{A}$.
- $\mathcal{A}$ sends two pairs of messages $((m_0^{\mathcal{B}}, m_1^{\mathcal{B}}), (m_0^{\mathcal{C}}, m_1^{\mathcal{C}}))$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- Ch samples $b \xleftarrow{\$} \{0,1\}$.
- Let $\mathsf{ct}^{\mathcal{B}}, \mathsf{ct}^{\mathcal{C}} \leftarrow \mathcal{D}(1^{\lambda}, b, \mathsf{pk})$.
- Apply $(\mathcal{B}(\mathsf{ct}^{\mathcal{B}}, \cdot) \otimes \mathcal{C}(\mathsf{ct}^{\mathcal{C}}, \cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.
- Output 1 if $b^{\mathcal{B}} = b_0$ and $b^{\mathcal{C}} = b_1$.

Figure 36: $\mathcal{D}$-CPA anti-piracy

A.3 Unclonable Encryption

An unclonable encryption scheme is a triple of QPT algorithms $\mathsf{UE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ given below:

- $\mathsf{Gen}(1^{\lambda}) : \mathsf{sk}$ on input a security parameter 1^{λ}, returns a classical key sk.
- $\mathsf{Enc}(\mathsf{sk}, m) : \rho_{ct}$ takes the key sk, a message $m \in \{0,1\}^{n(\lambda)}$ and outputs a quantum ciphertext ρ_{ct}.
- $\mathsf{Dec}(\mathsf{sk}, \rho_{ct}) : \rho_m$ takes a secret key sk, a quantum ciphertext ρ_{ct} and outputs a message m'.

Correctness. The following must hold for the encryption scheme. For every $m \in \{0,1\}^{n(\lambda)}$, the following holds:

$$\mathsf{Pr}\left[m \leftarrow \mathsf{Dec}(\mathsf{sk}, \rho_{ct}) \ \middle|\ \substack{\mathsf{sk} \leftarrow \mathsf{Gen}(1^{\lambda}) \\ \rho_{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, m)}\right] \geq 1 - \mathsf{negl}(\lambda)$$

CPA security. We say that an unclonable encryption scheme UE satisfies CPA security if for every QPT adversary $(\mathcal{A}, \mathcal{B}, \mathcal{C})$, there exists a negligible function negl such that

$$\mathsf{Pr}\left[1 \leftarrow \mathsf{UE.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C})}\left(1^{\lambda}\right)\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

B Related Work

Unclonable cryptography is an emerging area in quantum cryptography. The origins of this area date back to 1980s when Weisner [Wie83] first conceived the idea of quantum money which leverages

$\underline{\mathsf{UE.Expt}^{(\mathcal{A},\mathcal{B},\mathcal{C})}\left(1^\lambda\right)}$:

- Ch samples $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda)$.
- $\mathcal{A}$ sends a pair of messages (m_0, m_1).
- Ch picks a bit b uniformly at random. Ch generates $\rho_{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, m_b)$.
- $\mathcal{A}$ produces a bipartite state $\sigma_{\mathcal{B},\mathcal{C}}$.
- Apply $(\mathcal{B}(\mathsf{sk},\cdot) \otimes \mathcal{C}(\mathsf{sk},\cdot))(\sigma_{\mathcal{B},\mathcal{C}})$ to obtain $(b^{\mathcal{B}}, b^{\mathcal{C}})$.
- Output 1 if $b^{\mathcal{B}} = b^{\mathcal{C}} = b$.

Figure 37: CPA security

the no-cloning principle to design money states that cannot be counterfeited. Designing quantum money has been an active and an important research direction [Aar09, AC12, Zha19, Shm22, LMZ23, Zha23]. Since the inception of quantum money, there have been numerous unclonable primitives proposed and studied. We briefly discuss the most relevant ones to our work below.

Copy-Protection. Aaronson [Aar09] conceived the notion of quantum copy-protection. Roughly speaking, in a quantum copy-protection scheme, a quantum state is associated with functionality such that given this state, we can still evaluate the functionality while on the other hand, it should be hard to replicate this state and send this to many parties. Understanding the feasibility of copy-protection for unlearnable functionalities has been an intriguing direction. Copy-protecting arbitrary unlearnable functions is known to be impossible in the plain model [AL21] assuming cryptographic assumptions. Even in the quantum random oracle model, the existence of a restricted class of copy-protection schemes have been ruled out [AK22]. This was complemented by [ALL+21] who showed that any class of unlearnable functions can be copy-protected in the presence of a classical oracle. The breakthrough work of [CLLZ21] showed for the first time that copy-protection for interesting classes of unlearnable functions exists in the plain model. This was followed by the work of [LLQZ22] who identified some watermarkable functions that can be copy-protected. Notably, both [CLLZ21] and [LLQZ22] only focus on copy-protecting specific functionalities whereas we identify a broader class of functionalities that can be copy-protected. Finally, a recent work [CHV23] shows how to copy-protect point functions in the plain model. The same work also shows how to de-quantize communication in copy-protection schemes.

Unclonable and Single-Decryptor Encryption. Associating encryption schemes with unclonability properties were studied in the works of [BL20, BI20, GZ20]. In an encryption scheme, either we can protect the decryption key or the ciphertext from being cloned, resulting in two different notions.

In an unclonable encryption scheme, introduced by [BL20], given one copy of a ciphertext, it should be infeasible to produce many copies of the ciphertext. There are two ways to formalize the security of an unclonable encryption scheme. Roughly speaking, search security is defined as follows: if the adversary can produce two copies from one copy then it should be infeasible for two non-communicating adversaries $\mathcal{B}$ and $\mathcal{C}$, who receive a copy each, to simultaneously recover the entire message. Specifically, the security notion does not prevent both $\mathcal{B}$ and $\mathcal{C}$ from learning a few bits of the message. On the other hand, indistinguishability security is a stronger notion that disallows $\mathcal{B}$ and $\mathcal{C}$ to simultaneously determine which of m_0 or m_1, for two adversarially chosen messages (m_0, m_1), were encrypted. [BL20] showed that unclonable encryption with search security for long messages exists. Achieving indistinguishability security in the plain model has been left as an important open problem. A couple of recent works [AKL+22, AKL23] shows how to achieve indistinguishability security in the quantum random oracle model. Both [AKL+22, AKL23] achieve unclonable encryption in the one-time secret-key setting and this can be upgraded to a public-key scheme using the compiler of [AK21].

In a single-decryptor encryption scheme, introduced by [GZ20], the decryption key is associated with a quantum state such that given this quantum state, we can still perform decryption but on the other hand, it should be infeasible for an adversary who receives one copy of the state to produce two states, each given to $\mathcal{B}$ and $\mathcal{C}$, such that $\mathcal{B}$ and $\mathcal{C}$ independently have the ability to decrypt. As before, we can consider both search and indistinguishability security; for the rest of the discussion, we focus on indistinguishability security. [CLLZ21] first constructed single-decryptor encryption in the public-key setting assuming indistinguishability obfuscation (iO) and learning with errors. Recent works [AKL23] and [KN23] present information-theoretic constructions and constructions based on learning with errors in the one-time setting. The challenge distribution in the security of single-decryptor encryption is an important parameter to consider. In the security experiment, $\mathcal{B}$ and $\mathcal{C}$ each respectively receive ciphertexts $\mathsf{ct}_{\mathcal{B}}$ and $\mathsf{ct}_{\mathcal{C}}$, where $(\mathsf{ct}_{\mathcal{B}}, \mathsf{ct}_{\mathcal{C}})$ is drawn from a distribution referred to as challenge distribution. Most of the existing results focus on the setting when the challenge distribution is a product distribution, referred to as independent challenge distribution. Typically, achieving independent challenge distribution is easier than achieving identical distribution, which corresponds to the case when both $\mathcal{B}$ and $\mathcal{C}$ receive as input the same ciphertext. Indeed, there is a reason for this: single-decryptor encryption with security against identical challenge distribution implies unclonable encryption. In this work, we show how to achieve public-key single-decryptor encryption under identical challenge distribution.

C Additional Preliminaries

C.1 Indistinguishability Obfuscation (IO)

An obfuscation scheme associated with a class of circuit $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ consists of two probabilistic polynomial-time algorithms $\mathsf{iO} = (\mathsf{Obf}, \mathsf{Eval})$ defined below.

- **Obfuscate**, $C' \leftarrow \mathsf{Obf}(1^\lambda, C)$: takes as input security parameter λ, a circuit $C \in \mathcal{C}_\lambda$ and outputs an obfuscation of C, C'.
- **Evaluation**, $y \leftarrow \mathsf{Eval}(C', x)$: a deterministic algorithm that takes as input an obfuscated circuit C', an input $x \in \{0,1\}^\lambda$ and outputs y.

Definition 79 ([BGI+01]). *An obfuscation scheme* $\mathsf{iO} = (\mathsf{Obf}, \mathsf{Eval})$ *is a* **post-quantum secure indistinguishability obfuscator** *for a class of circuits* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$*, with every* $C \in \mathcal{C}_\lambda$ *has size poly*(λ)*, if it satisfies the following properties:*

- **Perfect correctness:** *For every* $C : \{0,1\}^\lambda \to \{0,1\} \in \mathcal{C}_\lambda$*,* $x \in \{0,1\}^\lambda$ *it holds that:*

$$\Pr\left[\mathsf{Eval}\big(\mathsf{Obf}(1^\lambda, C), x\big) = C(x)\right] = 1 \ .$$

- **Polynomial Slowdown:** *For every* $C : \{0,1\}^\lambda \to \{0,1\} \in \mathcal{C}_\lambda$*, we have the running time of* Obf *on input* $(1^\lambda, C)$ *to be poly*$(|C|, \lambda)$*. Similarly, we have the running time of* Eval *on input* (C', x) *is* $\mathsf{poly}(|C'|, \lambda)$

- **Security:** *For every QPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mu(\cdot)$*, such that for every sufficiently large* $\lambda \in \mathbb{N}$*, for every* $C_0, C_1 \in \mathcal{C}_\lambda$ *with* $C_0(x) = C_1(x)$ *for every* $x \in \{0,1\}^\lambda$ *and* $|C_0| = |C_1|$*, we have:*

$$\left|\Pr\left[\mathcal{A}\big(\mathsf{Obf}(1^\lambda, C_0), C_0, C_1\big) = 1\right] - \Pr\left[\mathcal{A}\big(\mathsf{Obf}(1^\lambda, C_1), C_0, C_1\big) = 1\right]\right| \leq \mu(\lambda) \ .$$