Unclonable Cryptography with Unbounded Collusions and Impossibility of Hyperefficient Shadow Tomography

Alper Çakan
Carnegie Mellon University
acakan@cs.cmu.edu

Vipul Goyal
NTT Research & Carnegie Mellon University
vipul@vipulgoyal.org

Abstract

Quantum no-cloning theorem gives rise to the intriguing possibility of quantum copy protection where we encode a program or functionality in a quantum state such that a user in possession of k copies cannot create $k+1$ copies, for any k. Introduced by Aaronson (CCC'09) over a decade ago, copy protection has proven to be notoriously hard to achieve. Previous work has been able to achieve copy-protection for various functionalities only in restricted models: (i) in the bounded collusion setting where $k \to k+1$ security is achieved for a-priori fixed collusion bound k (in the plain model with the same computational assumptions as ours, by Liu, Liu, Qian, Zhandry [TCC'22]), or, (ii) only $k \to 2k$ security is achieved (relative to a structured quantum oracle, by Aaronson [CCC'09]).

In this work, we give the first *unbounded* collusion-resistant (i.e. multiple-copy secure) copy-protection schemes, answering the long-standing open question of constructing such schemes, raised by multiple previous works starting with Aaronson (CCC'09).

More specifically, we obtain the following results.

- We construct (i) public-key encryption, (ii) public-key functional encryption, (iii) signature and (iv) pseudorandom function schemes whose keys are copy-protected against unbounded collusions in the plain model (i.e. without any idealized oracles), assuming (post-quantum) subexponentially secure $i\mathcal{O}$ and LWE.
- We show that any unlearnable functionality can be copy-protected against unbounded collusions, relative to a classical oracle.
- As a corollary of our results, we rule out the existence of *hyperefficient quantum shadow tomography*,
  - even given non-black-box access to the measurements, assuming subexponentially secure $i\mathcal{O}$ and LWE, or,
  - unconditionally relative to a quantumly accessible *classical* oracle,

  and hence answer an open question by Aaronson (STOC'18).

We obtain our results through a novel technique which uses identity-based encryption to construct multiple copy secure copy-protection schemes from $\mathsf{1\text{-}copy} \to \mathsf{2\text{-}copy}$ secure schemes. We believe our technique is of independent interest.

Along the way, we also obtain the following results.

- We define and prove the security of new collusion-resistant monogamy-of-entanglement games for *coset states*.
- We construct a classical puncturable functional encryption scheme whose master secret key can be punctured at all functions f such that $f(m_0) \neq f(m_1)$. This might also be of independent interest.

Keywords: Quantum cryptography, copy-protection, unclonable cryptography, shadow tomography

Contents

1 Introduction

The no-cloning principle, a fundamental implication of quantum mechanics, shows that arbitrary unknown quantum states cannot be copied. This simple principle allows us to imagine applications that are classically impossible. Indeed, it has found a wide range of applications in cryptography, starting with the work of Wiener [Wie83] where he puts forward the notion of *quantum money*, where we imagine that there is a bank producing quantum states, called *banknotes*, that are secure against counterfeiting: any (malicious) user in possession of k banknotes for any k cannot produce $k+1$ *authentic* banknotes. The interesting notion of quantum banknotes (i.e., unclonable authenticatable quantum states) also led Aaronson [Aar09] to pose the following question:

> Can we use quantum information to *copy-protect functionalities/programs*, where user(s) in possession of some number of copies of a program P cannot produce more working copies?

In more detail, we want to achieve the following. A vendor encodes a functionality[1] into a quantum state, and a user in possession of such a state can use it to evaluate the functionality any number of times, and we want to achieve $a \to b$ copy-protection: any malicious user(s) in possession of a such copies of the program cannot produce b working copies. Similar to quantum money, this is an impossible feat in a classical world since classical information can be readily copied any amount of times. Therefore, in a classical world, once you are given a single working copy of the program, you can make any number of copies of it.

Perhaps surprisingly, [Aar09] showed copy-protection using quantum information is indeed possible: relative to a *structured*[2] *quantum oracle*, any *unlearnable* program can be copy-protected in a way that is $k \to k+r$ secure (for any [polynomial] k and some $r > k$). That is, in the construction of [Aar09], the adversary is prevented from doubling their number of working copies. Later, Aaronson et al. [ALL+21] showed that relative to a *classical* structured oracle (that depends on the program being copy protected) model, any unlearnable program can be copy-protected, but this time only in the $\mathsf{1\text{-}copy} \to \mathsf{2\text{-}copy}$ setting.

Copy-Protecting Decryption Keys, PRFs and Signing Keys In a related line of work, Georgiou and Zhandry [GZ20] started the study of *single-decryptor encryption*, that is, copy-protection for decryption functionality (i.e. secret keys) of a public-key encryption (PKE) scheme where an adversary tries to create $k+1$ working decryption keys given only k copy-protected keys. More formally, in this model, a *pirate* adversary obtains the classical public key and k copy-protected quantum secret keys of the scheme. Then, it produces $k+1$ *freeloader* adversaries that are possibly entangled but not communicating[3], and these freeloaders are presented with classical challenge ciphertexts. We require that they cannot all succeed in decrypting simultaneously. [GZ20] also gave a $1 \to 2$ secure copy-protection scheme relative to a structured oracle. Later, Coladangelo et al. [CLLZ21] showed how to construct a $1 \to 2$ copy-protected public-key encryption using *coset states*, this time in the plain model, assuming quantum hardness of LWE, (post-quantum) subexponentially secure indistinguishability obfuscation and one-way functions. They also construct

[1] For example, a proprietary software or a decryption program/key of an encryption system that is used to distribute encrypted content

[2] The oracle used in this construction takes as input a function and a value, evaluates the function on the value, or takes as input a function and outputs a Haar random state associated with it.

[3] If they were allowed to communicate, one freeloader could hold the secret key and all the other freeloaders would simply send their challenge ciphertexts to him to decrypt and send back the result.

$1 \to 2$ copy-protection schemes for pseudorandom functions (PRF), based on the same assumptions. Liu et al. [LLQZ22] constructed *bounded* collusion-resistant PKE and PRF schemes, by showing through an elegant proof that the k-way parallel repetitions of the schemes of [CLLZ21] are bounded $k \to k+1$ copy-protection secure. Further, they also construct a *bounded* $k \to k+1$ copy-protection secure scheme for the signing keys of a signature scheme. However, for all schemes of [LLQZ22], the collusion-bound k is fixed during setup, the sizes of the schemes grow (linearly) with the bound k and the copy-protected key generation is stateful.

Collusion-Resistant Copy-Protection Unfortunately, none of the previous work satisfy the most-general notion of unbounded collusion-resistant copy-protection where we require $k \to k+1$ security for all polynomials k (that is not known and hence the size of the scheme does not depend on it).

In particular, all schemes of [ALL$^+$21], [CLLZ21] and [LLQZ22] can easily be broken when the adversary obtains multiple copies. Any 2 users (in case of the first two works) or $k+1$ users (for the fixed k value, in case of [LLQZ22])[4] with copy-protected keys can create an *anonymous classical* program/string (which can be copied/distributed any number of times) that can be used to decrypt any ciphertext in case of encryption schemes, or evaluate/sign any input in case of general programs, PRFs and signatures. The only other scheme, that of [Aar09], is only $k \to 2k$ secure rather than $k \to k+1$, and more importantly, since it relies on structured quantum oracles, it cannot even be heuristically instantiated since we do not have any (even candidate) constructions of general-purpose quantum circuit obfuscation. In fact, [LLQZ22] argues that even any extension of the scheme of [Aar09] would require such obfuscation, since it uses Haar random states and there is evidence that these states cannot be *classically verified* ([LLQZ22, Kre21]).

We believe that the security guarantees of the previous work ([ALL$^+$21], [CLLZ21], [LLQZ22]) are very unrealistic in the age of the Internet: the users can actually mount the anonymous attacks described above through classical channels, by simply measuring their key and sending the classical measurement result to other parties or posting it online!

Computational Complexity of Shadow Tomography Lastly, aside from theoretical interest in the unbounded collusion setting in and of itself, we note that it is a theoretically important problem also due to its intimate connection to the computational complexity of another important problem, *shadow tomography* [Aar18] (see Section 1.1 and Section 12).

The above state of affairs leaves open the following natural question also raised explicitly in several previous works [Aar09, AC12, CLLZ21, LLQZ22]:

Can we use quantum information to construct unbounded collusion-resistant
copy-protection schemes?

In this work, we answer the above question positively, in the plain model, with computational assumptions matching the previous work.

1.1 Our Results

In this work, we resolve the long-standing open problem of constructing fully collusion-resistant copy-protection schemes by constructing such schemes for public-key encryption, public-key functional encryption, signatures and pseudorandom functions, all in the plain model.

[4] We re-emphasize that the size of the scheme (e.g. ciphertext and public-key sizes) grows with the set k value, so it cannot be set arbitrarily large.

Copy-Protecting Decryption Keys (Section 7 and Section 8) We construct encryption schemes where the secret keys are copy-protected.

Theorem 1. *Assuming post-quantum subexponentially secure indistinguishability obfuscation and subexponentially secure LWE, there exists a public-key encryption scheme with fully collusion-resistant copy-protected secret keys.*

Our computational assumptions above match[5] the assumptions made by [CLLZ21] to achieve 1 → 2 copy-protection and those made by [LLQZ22] to achieve $k \to k{+}1$ bounded collusion-resistant copy-protected public-key encryption schemes.

Theorem 2. *Assuming post-quantum subexponentially secure indistinguishability obfuscation and subexponentially secure LWE, there exists a public-key* functional *encryption scheme with fully collusion-resistant copy-protected secret keys.*

Prior to our work, the only construction of functional encryption with copy-protected secret keys (given by Kitagawa and Nishimaki [KN22]) was in the 1 → 2 copy-protection setting, based on assumptions same as ours, and in a weaker security model where no key queries were allowed after seeing the challenge ciphertext (see Section 2.6 for more details). Furthermore, on top of matching the assumptions previous work used for constructing copy-protected public-key encryption, the $i\mathcal{O}$ assumption we make for our copy-protected FE scheme can be considered necessary since functional encryption is known to be equivalent to indistinguishability obfuscation (up to subexponential security loss) [BV18].

Since functional encryption can be used to construct identity-based encryption [Sha85] and attribute-based encryption [SW05, GPSW06] in a straightforward manner, our work also gives the first identity-based encryption and attribute-based encryption schemes with collusion-resistant copy-protected secret keys. Through copy-protected identity-based encryption, we can also obtain *unclonable identity cards*, first suggested by [Aar09].

Copy-Protecting PRF and Signature Keys (Section 10 and Section 9) We also construct copy-protection schemes for a family of pseudorandom functions (PRF) and signing keys of a signature scheme.

Theorem 3. *Assuming post-quantum subexponentially secure indistinguishability obfuscation and subexponentially secure LWE, there exists a PRF and a signature scheme with fully collusion-resistant copy-protected keys.*

Copy-Protecting All Unlearnable Functionalities We also show how to copy-protect any unlearnable functionality, relative to a classical oracle.

Theorem 4. *Assuming post-quantum subexponentially secure one-way functions*[6]*, for any unlearnable functionality, there exists a fully collusion-resistant copy-protection scheme relative to an* efficient *classical oracle.*

[5] More specifically, our assumptions exactly match the assumptions made by [LLQZ22], but [CLLZ21] assumes polynomially secure LWE whereas we assume subexponentially secure LWE. We emphasize that [CLLZ21] still assume subexponentially secure $i\mathcal{O}$ and subexponentially secure one-way functions.

[6] We can also achieve this result unconditionally, if we do not insist that the classical oracle is efficient.

This supersedes[7] both [Aar09], which uses a structured quantum oracle and only satisfies $k \to 2k$ copy-protection, and [ALL+21] which uses a structured classical oracle but only satisfies $1 \to 2$ copy-protection.

Impossibility of Hyperefficient Shadow Tomography (Section 12) Shadow tomography, introduced by Aaronson [Aar18], is the following task: Given many copies of a mixed state ρ and a list of binary measurements $\{E_1, \dots, E_M\}$, estimate the acceptance probabilities $\text{Tr}(E_i\rho)$ of measurements E_i within additive error ε, for all measurements $i \in [M]$. This task has important ramifications for quantum information theory, since it means that we can learn many properties of a quantum state without needing to do a full tomography of it, which necessarily requires exponentially many copies of the state [OW16]. It has also found many applications in cryptography, such as (i) [Aar18] who shows that unconditional copy-protection is not possible (ii) [BGHD+23] who shows that unconditional PKE cannot exist even if we allow public-keys to be quantum and (iii) [KT24] who shows that unconditional one-way state generators cannot exist. Lastly, shadow tomography also has connections to the question of *classical vs. quantum advice*, and the related complexity classes $\mathsf{BQP/poly}$ and $\mathsf{BQP/qpoly}$. Note that in general, and in particular in all of these applications, the measurement set is indexed by all possible strings in some support and M is exponential in the security parameter or in the number of qubits. In fact, the case $M = \mathsf{poly}(\lambda)$ can be trivially solved in polynomial (M/ε^2) time with polynomially many copies, by estimating each $\text{Tr}(E_i\rho)$ for $i \in [M]$ simply by actually performing the measurements E_i multiple times on separate copies.

[Aar18] showed that shadow tomography can be performed in a sample-efficient manner; using $\mathsf{poly}(n, \log M, \frac{1}{\varepsilon})$ copies of an n-qubit state ρ, however, their scheme is not *computationally efficient*, with time complexity $\tilde{O}(M)$[8]. In light of above, they posed the following as an open question: is *hyperefficient* shadow tomography possible? That is, is it possible to perform shadow tomography with time complexity $\mathsf{poly}(n, \log M, \frac{1}{\varepsilon})$? Note that in this case, we ask that the set of measurements $\{E_i\}_{i \in M}$ be implemented by a uniform quantum algorithm E that on input i, τ, applies the measurement E_i to the state τ. We will be given this quantum circuit E as input and we are asked to output a quantum circuit C such that $C(i)$ estimates $\text{Tr}(E_i\rho)$ for all i.[9]

Previously, hyperefficient shadow tomography was ruled out only relative to quantum oracles [Aar18, AK07, Kre21], where we only get oracle access to the measurement circuit E. Through a generic attack on copy-protection schemes using shadow tomography given by [Aar18, SW22], a corollary of our results is the impossibility of hyperefficient shadow tomography, answering the open question of [Aar18].

Corollary 1. *Assuming post-quantum subexponentially secure indistinguishability obfuscation and LWE, there does not exist a hyperefficient shadow tomography algorithm.*

Corollary 2. *Assuming post-quantum subexponentially secure one-way functions*[10]*, relative to an efficient* classical *oracle, there cannot exist a hyperefficient shadow tomography algorithm.*

[7] Note that Theorem 4 and similar results of [Aar09, ALL+20] cannot be securely instantiated in the plain model for all unlearnable functionalities, since [ALP21] proves that there exists an unlearnable functionality that cannot be copy-protected in the plain model.

[8] As noted above, M is exponential in the security parameter or in the number of qubits.

[9] Without these assumptions, even reading the descriptions of all measurements or outputting all the estimates would take $\Omega(M)$ time.

[10] Similar to before, we can achieve this result unconditionally if we do not insist on efficient oracles

We note that making computational assumptions is necessary, since, hyperefficient shadow tomography is possible given access to PP oracle.[11]

Technical Contributions and Additional Results An important contribution of our work is a novel technique to construct collusion-resistant copy-protection schemes which relies on using identity-based encryption (IBE). We use this technique in all of our constructions and we believe it to be of independent interest. Our technique could be considered an analogue of the technique of using digital signatures to construct full-fledged (i.e. collusion-resistant) quantum money from single banknote schemes [LAF+09, FGH+12, AC12]. We also define and prove the security of new collusion-resistant *monogamy-of-entanglement* games [CLLZ21, CV22] for coset states to prove the security of our schemes. See Section 5 for details.

Finally, using the techniques we employ to prove the security of our functional encryption scheme, we also give a construction of a classical functional encryption scheme where the master secret key can be punctured such that the resulting master key allows issuing keys only for functions f that satisfy $f(m_0) = f(m_1)$. This allows us to remove the interaction/key queries after the challenge ciphertext in the usual functional encryption security game (Definition 6), since the adversary can issue their own keys using the punctured master secret key. This might also be of independent interest. See Section 6.4 for details.

Theorem 5. *Assuming subexponentially secure indistinguishability obfuscation and one-way functions, there exists a functional encryption scheme whose master secret key can be punctured at all functions f such that $f(m_0) \neq f(m_1)$.*

2 Technical Overview

2.1 Public-Key Encryption with Copy-Protected Secret Keys

Let us first describe our security model, which is the same as previous work [Aar09, GZ20, CLLZ21, LLQZ22]. We consider a public-key encryption scheme with classical ciphertexts, a classical public-key and an additional (quantum) algorithm $\mathsf{QKeyGen}$. The copy-protected key generation algorithm $\mathsf{QKeyGen}$, on input the classical secret key, outputs a reusable quantum state that can be used to decrypt any number of times. For security, we will require that a user with k copy-protected secret keys cannot create $k+1$ keys. More formally, in an *anti-piracy game* (Definition 24) for public-key encryption, we have an adversary, called a *pirate*. This adversary is given the public key pk, and then for any (polynomial) number of rounds, it queries for quantum copy-protected secret keys. After it is done, it outputs pairs of challenge messages $(m_\ell^0, m_\ell^1)_{\ell \in [k+1]}$ and $k+1$ (possibly entangled) *freeloader* adversaries, where k is the number of copy-protected keys it has queried. Then, the challenger samples challenge bits b_ℓ, and presents each freeloader with $\mathsf{Enc}(pk, m_\ell^{b_\ell})$. The freeloaders output their predictions b'_ℓ, and the adversary wins if $b'_\ell = b_\ell$ for all $\ell \in [k+1]$. We require that no efficient adversary can win with probability better than $1/2 + \mathsf{negl}(\lambda)$. The baseline success probability is $1/2$, since the pirate adversary can output k of its keys to the first k freeloaders, and let the last freeloader randomly guess the challenge bit b_{k+1}.

1 → 2 **Copy-Protection Secure Construction of Coladangelo et al. [CLLZ21]**

As a warm-up, we will recall the 1 → 2 copy-protection secure construction based on coset states, given by [CLLZ21], which also forms the base of our construction.

[11] We thank an anonymous reviewer for pointing out this remark.

A coset state [CLLZ21, VZ21] is a state of the form $\sum_{a\in A}(-1)^{\langle s',a\rangle}|a+s\rangle =: |A_{s,s'}\rangle$ where $A \subseteq \mathbb{F}_2^n$ is a subspace and $s, s' \in \mathbb{F}_2^n$. [CLLZ21, CV22] showed that coset states satisfy a property called *strong monogamy-of-entanglement (MoE)*, which is as follows. Consider the following game between an adversary tuple $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2$ and a challenger. Challenger uniformly at random samples a subspace $A \subseteq \mathbb{F}_2^n$ of dimension $n/2$ and elements $s, s' \in \mathbb{F}_2^n$, and submits $|A_{s,s'}\rangle$ and the obfuscated programs[12] $i\mathcal{O}(A+s), i\mathcal{O}(A^\perp + s')$ to the adversary $\mathcal{A}_0$. Then, the adversary $\mathcal{A}_0$ outputs two (entangled) registers $\mathsf{R}_1, \mathsf{R}_2$, for $\mathcal{A}_1, \mathcal{A}_2$. Then, $\mathcal{A}_1, \mathcal{A}_2$ receive their registers and also the description of the subspace A (but not the vectors s, s' of course). Finally, $\mathcal{A}_1$ is required to output a vector in $A+s$ and $\mathcal{A}_2$ is required to output a vector in $A^\perp + s'$. Strong MoE property says that no efficient adversary can win this game with non-negligible probability. In a variation used implicitly by [CLLZ21] and later formalized in a different context by [ÇGLZR24], we present $\mathcal{A}_0$ with multiple, say c many, independent coset states (called a *coset state tuple*) and the corresponding membership checking programs, and require that $\mathcal{A}_1, \mathcal{A}_2$ each output vectors in $A_i + s_i$ or $A_i^\perp + s_i'$ for all $i \in [c]$, depending on random challenge strings $r^1, r^2 \in \{0,1\}^c$ presented to them. By a reduction to the original version, it can be shown that no efficient adversary can win this game with non-negligible probability (Theorem 23). We call this variation the multi-challenge version.

Now, we move onto the copy-protected public-key encryption construction of [CLLZ21]. During setup, we sample a coset tuple $(A_i, s_i, s_i')_{i\in[c]}$. The coset state tuple $\left|A_{i,s_i,s_i'}\right\rangle_{i\in[c(\lambda)]}$ becomes the copy-protected quantum secret key, and we output $pk = (i\mathcal{O}(A_i+s), i\mathcal{O}(A_i^\perp + s_i'))_{i\in[c(\lambda)]}$ as the public key. Finally, to encrypt a message m, we sample a random string r and an indistinguishability obfuscation $\mathsf{OP} \leftarrow i\mathcal{O}(\mathsf{PCt}_{pk,r,m})$, where $\mathsf{PCt}_{pk,r,m}$ is a program that takes is input vectors $(v_i)_{i\in[c]}$ and, checks if they are in correct cosets with respect to r. That is, we require $v \in A_i + s_i$ if the i-th bit of r is 0 and $v \in A_i^\perp + s_i'$ if it is 1. The program $\mathsf{PCt}_{pk,r,m}$ outputs the message m if and only if the vectors pass the test. We output (OPCt, r) as the ciphertext. To decrpyt a message, we simply apply QFT (quantum Fourier transform) to our coset state tuple at indices where $(r)_i = 1$. Then, it is easy to see that running OPCt coherently on our key and measuring the result gives us m with probability 1[13].

On a high level, the security follows by multi-challenge MoE game, since the two freeloaders, to decrypt their ciphertexts, must be querying the programs $\mathsf{PCt}^{(1)}, \mathsf{PCt}^{(2)}$ at the correct vectors with respect to r_1, r_2 respectively, which is exactly the challenge in the MoE game. The proof is more involved since (i) $i\mathcal{O}$ is used rather than ideal oracles and (ii) the freeloaders can be entangled. We discuss this further in the upcoming sections.

Challenges for Collusion-Resistant Copy-Protection

First, we note that the construction of [CLLZ21] is trivially insecure when the adversary is given two copies of the secret key: The adversary can measure one copy of the state $\left|A_{i,s_i,s_i'}\right\rangle$ in the computational basis and the other copy in the Hadamard basis, thus obtaining vectors $v_i \in A_i + s_i$ and $w_i \in A_i^\perp + s_i'$ for all $i \in [c]$. Using these vectors, one can decrypt any ciphertext and since these vectors are classical information, the pirate adversary can indeed produce any number of working secret keys. Thus, the scheme only satisfies 1 → 2 unclonability.

One natural solution, argued by [LLQZ22], is to try and employ quantum states that already possess a collusion-resistant unclonability guarantee, such as Haar random states or their computa-

[12] Here, we overload the notation to let $A+s$ also denote the program that takes as input a vector v and outputs 1 if $v \in A+s$, and 0 if not, and similarly for $A^\perp + s$.

[13] By Gentle Measurement Lemma (Lemma 2), this also means that we can revert the quantum key back to its original state after decrypting a ciphertext.

tional neighbor, pseudorandom states. This is indeed the approach employed by [Aar09] to achieve $k \to 2k$ copy-protection relative to a structured quantum oracle. However, the problem is that there is no known way of verifying such states or employing these states to construct a copy-protection scheme without the use of quantum oracles, and there is evidence that this is an inherent property of such states [LLQZ22, Kre21].

Another natural solution, used by [LLQZ22], is to *independently* sample a new coset state tuple $\left|A^{(j)}_{i,s_i,s'_i}\right\rangle_{i\in[c(\lambda)]}$ whenever a copy-protected secret key is requested rather than giving out the same key state multiple times. In this case, the ciphertext program also takes as input the index j of the key the decryption procedure is using, and verifies the input vectors with respect to that coset tuple. Therefore, they need to include the corresponding obfuscated membership checking programs for each possible key in the public-key, since otherwise the ciphertexts would not be decryptable by that key. Therefore, we can only have k different key states for a fixed k chosen during setup (which is when pk is created). Therefore, the construction of [LLQZ22] only achieves $k \to k+1$ copy-protection where the collusion-bound k needs to be known at the time of setup, and the size of the scheme (public key, ciphertexts) grows with k, since the scheme basically consists of k independent instances of the $1 \to 2$ secure scheme of [CLLZ21]. Furthermore, similar to the scheme of [CLLZ21], this scheme becomes trivially insecure once given $k+1$ keys, since we will have obtained one of the coset state tuples twice.

Our Solution: Pseudorandom Coset States and Identity-Based Encryption

As discussed above, if we are sampling independent coset states for each copy-protected key query, we need to have an a-priori bound on the number of different keys. In the unbounded setting, since there are exponentially many cosets, it is not possible to verify all possible cosets using a polynomial size public key pk.

Our solution to this is to *compress* the public-key by using pseudorandom coset states rather than truly random ones. We sample a PRF key K and include it in the classical secret key. Then, whenever we need to sample a copy-protected quantum secret key using our classical secret key, we sample a random identity string id from $\{0,1\}^\lambda$ and then sample a coset state tuple using the randomness $F(K, id)$. Our public-key will be an obfuscated program OPMem_K (with PRF key K embedded) that takes in an id, some vectors $(v_i)_{i\in[c]}$ and a *basis* r, and verifies the vectors $(v_i)_{i\in[c]}$ with respect to r and the coset tuple associated with id. We now have a polynomial size public-key that allows us to verify any possible (honest) coset state tuple.

A high level intuition for security is as follows, where for now we assume we use ideal oracles instead of $i\mathcal{O}$. By PRF security, the adversary's view is indistinguishable from having obtained k independent coset state tuples since for any efficient adversary that obtains any (polynomial) number of quantum secret keys, they will all have unique identity strings with overwhelming probability. Note that we still need to argue that one cannot produce $k+1$ working keys from k independent coset state tuples, which we discuss how to argue in Section 2.2.

However, in reality, we are using $i\mathcal{O}$ and not ideal oracles. Now, the first problem is that, the coset state tuples that the adversary obtains during key query phase are no longer pseudorandom, since the adversary does not only have query access to the PRF but rather has the PRF key K inside pk. A standard solution when using PRFs and indistinguishability obfuscation is to puncture the PRF key at some inputs. Let $id_1, \ldots, id_k$ be the identity strings of the k copy-protected keys obtained by the adversary. We can try to puncture the PRF key at $id_1, \ldots, id_k$, but this would make the size of our public-key dependent on k. A much more important problem is that the adversary is not required to run PCt on only one of id_i, and in fact, PCt might be leaking[14] m. Or, the adversary somehow might be obtaining the hidden message m by running it on some unrelated identity id and vectors that pass the verification of PMem for id[15]. The latter is because the adversary has access to K in some form (i.e. inside PMem), therefore, it might be somehow obtaining $F(K, id)$ for some id. To rule this possibility out, we would need to puncture the PRF key at all strings in $\{0,1\}^\lambda$!

To solve this problem and to puncture the PRF key only at few points, we first want to make sure that the adversary can obtain the hidden message m only by running PCt on an identity string associated with one of the copy-protected keys it did obtain. To ensure this, we use the following approach based on identity-based encryption (IBE)(Definition 15). When PCt is queried on some id and some vectors $(u_j)_j$, after verifying that the vectors are in the correct cosets with respect to id and r, the program PCt outputs an IBE encryption of m under the identity id, rather than m in the clear. We will also change our copy-protected key generation algorithm to output the IBE secret key associated with id. Now, we will be able to argue that if an adversary is able to decrypt a ciphertext and obtain m, then it must have obtained $\mathsf{IBE.Enc}(pk, id_i, m)$ for some id_i. This is because by the security of IBE, the adversary cannot decrypt ciphertexts under identities other than $id_1, \dots, id_k$ - the only identities for which it has obtained the IBE secret keys. Above in turn means that the adversary must have run PCt on id_i and the correct vectors for the coset tuple associated with id_i. In essence, we are forcing the adversary the clone one of the original copy-protected secret keys rather than coming up with a new key. Hence, we will eventually reduce to the MoE security of the coset state tuple associated with some id_i. Now, we need to only puncture the PRF key at (at most) k points! This is still too many.[16] However, we observe the following: the adversary obtains k secret keys $sk_{id_1}, \dots, sk_{id_k}$ of the IBE scheme while there are $k+1$ freeloaders. Hence, by pigeonhole principle, two of the $k+1$ freeloaders must be using the same key sk_{id_i} for some $i \in [k]$, and hence, the same coset state tuple - the one associated with id_i. As a result, we will only need to puncture the PRF key K at id_i. See Section 7.2 for the full scheme.

2.2 Proving Security

In this section, we give a high-level overview of the security proof of our public-key encryption construction. Our goal is to reduce the security of our scheme to the monogamy-of-entanglement game (see Section 2.1 and Theorem 24), which we will do so by extracting coset vectors $(v_i)_{i\in[c]}$ from the freeloader adversaries. On a high level, our proof uses ideas from [CLLZ21, ALL+21] for simultaneous extraction from entangled adversaries. The security proof of our functional encryption construction follows similarly and we refer the reader to Section 8 for details.

Note that in general, applying an extraction (which is essentially a measurement) on one of the freeloader adversaries might irreversibly damage the other ones since they are entangled. We will first make the testing of the freeloaders projective, which will allow us to argue that we can extract vectors from entangled adversaries since (i) repeating a projective measurement always gives the same outcome and does not change the state, (ii) acting (e.g. extracting) on some part of a state, informally, does not change the behaviour of projective measurements on the other part *too much* (Theorem 9). Now, let us briefly discuss *projective implementations*, introduced by Zhandry [Zha20]. Let $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_0 = I - \mathcal{E}\}$ be a binary POVM. [Zha20] shows that there is a *projective*

[14] Since we are not using black-box obfuscation for PCt.

[15] Since we are not using black-box obfuscation for PMem.

[16] Remember that when obfuscating a program using $i\mathcal{O}$, all programs that we will move between must be of the same size. Thus, if we are puncturing the PRF key at k points, our initial obfuscated public-key program needs to be padded to a size that depends on k.

measurement (indexed by a finite subset of $\mathbb{R}_{0\leq\cdot\leq 1}$) denoted $\mathsf{PI}(\mathcal{E})$ such that the following procedure has the same output distribution as applying $\mathcal{E}$ to ρ, for any state ρ.[17]

1. Apply $\mathsf{PI}(\mathcal{E})$ to ρ obtain a value $p \in [0, 1]$.
2. Output 1 with probability p.

Essentially, the projective implementation estimates the probability that $\mathcal{E}$ *accepts* ρ, and does so through a projective measurement. Note that $\mathsf{PI}(\mathcal{E})$ in general is inefficient, however, it can be approximated efficiently [Zha20]. We will ignore this issue in this section - see Section 7.3 for details.

In our anti-piracy game (Definition 24), we assume that the pirate adversary outputs each freeloader as (U, σ) where U is a unitary and σ is some quantum state. We interpret this as a quantum circuit[18] (with some hardwired quantum state) that takes in a challenge ciphertext and outputs a prediction b'. The challenger executes the freeloader using an appropriate universal quantum circuit. Now, let $\mathcal{D}$ be a ciphertext distribution and let (U_i, R_i) be a freeloader output by the pirate adversary (where R_i denotes the register containing the quantum part), and consider the following measurement on R_i.

1. Sample $b \leftarrow \{0, 1\}$.
2. Sample $ct \leftarrow \mathcal{D}(m_i^b)$.
3. Execute $U(ct_i, \mathsf{R}_i)$, measure the first qubit of the output registers in computational basis to obtain b'.
4. Output 1 if $b' = b$.

When we set $\mathcal{D}$ to be the honest ciphertext distribution where we encrypt m as $\mathsf{PKE.Enc}(pk, m)$, we see that the above measurement exactly corresponds to the testing of the freeloader in the anti-piracy game. Now, consider a modified game (parameterized by some inverse polynomial $\gamma(\lambda)$) where instead of performing this measurement directly, the challenger performs its projective implementation $\mathsf{PI}_{\mathcal{D}}$, and the adversary is said to win if the output is $> 1/2 + \gamma(\lambda)$ for all $k + 1$ freeloaders. Essentially, we are estimating the success probabilities of the freeloaders and comparing it to the baseline. Note that since $\mathsf{PI}_{\mathcal{D}}$ is projective, once we apply it and obtain a value p, the post-measurement state will again give p when its tested again for $\mathcal{D}$. [CLLZ21] proves that this modified game is stronger: it implies the security of the original anti-piracy game. Hence, we will prove security with respect to this stronger game.[19]

Now, we move onto a sketch of the security proof of our scheme. The idea is to test the freeloaders with respect to multiple challenge ciphertext distributions to pinpoint two freeloaders that use the same coset state tuple, and then extracting coset vectors from them and violating its $1 \to 2$ MoE security. Let us assume that an adversary wins the (modified) anti-piracy game (with probability $1/p(\lambda)$ where $p(\cdot)$ is a polynomial), meaning that applying $\mathsf{PI}_{\mathcal{D}}$ yields $> 1/2+\gamma(\lambda)$ for all $k+1$ freeloaders simultaneously with probability $> 1/p(\lambda)$. We define ciphertext distributions $\mathcal{D}_j$, for all $j \in \{0, 1, \dots, 2^\lambda\}$, representing all possible identity strings in $\{0, 1\}^\lambda$ (plus, the dummy upper

[17] We can equivalently say that the expected value of $\mathsf{PI}(\mathcal{E}) \cdot \rho$ is $\mathrm{Tr}[\mathcal{E}_1 \rho]$

[18] Note that while U is a unitary, this definition is enough to capture general quantum circuits since the adversary can also include empty workspace qubits inside σ, along with some quantum information obtained from the copy-protected keys.

[19] There is a caveat here that we need to prove security with respect to this game for all inverse polynomial $\gamma(\lambda)$ so that it implies security with respect to the original game.

bound 2^λ). We define $\mathcal{D}_j$ so that an encryption of a message m is $(i\mathcal{O}(\mathsf{PCt}^j), r)$ where PCt^j is the program that works as the honest ciphertext program if the input id satisfies $id \geq j$, and otherwise it replaces its hardcoded message m with $\top$ at the beginning. Observe that $\mathcal{D}_0$ corresponds to the honest ciphertext distribution, since $id < 0$ is never satisfied. Similarly, $\mathcal{D}_{2^\lambda}$ corresponds to the dummy ciphertext distribution where the message is not actually contained in the ciphertext. Now, consider the following thought experiment. We apply the measurements $\mathsf{PI}_{\mathcal{D}_i}$ sequentially from $j = 0$ to $j = 2^\lambda$, to all $k+1$ freeloaders. Let $q_{\ell,j}$ denote the outcomes for each freeloader $\ell \in [k+1]$. Intuitively, a non-negligible jump/gap between $q_{\ell,j}$ and $q_{\ell,j+1}$ for $j \in \{0, \dots, 2^\lambda - 1\}$ will mean that the freeloader ℓ is querying the ciphertext program at some vectors that are correct for the coset tuple associated with j. Since $\mathcal{D}_0$ is the honest ciphertext distribution of this scheme, the step $j = 0$ corresponds to the original security game and hence we get $q_{\ell,0} > 1/2 + \gamma$ for all $\ell \in [k+1]$ by assumption. We will also have $q_{\ell,2^\lambda} \leq 1/2$ for all $\ell \in [k+1]$ since the step $j = 2^\lambda$ corresponds to the ciphertext distribution $\mathcal{D}_{2^\lambda}$ that does not actually contain the message, and therefore no freeloader[20] can succeed with probability better than $1/2$ against $\mathcal{D}_{2^\lambda}$. Previous works ([AC12, LLQZ22]) use a pigeonhole principle to reduce $k \to k+1$ security to $1 \to 2$ unclonability security, where they conclude that two freeloaders must have a large gap between $|q_{\ell,j} - q_{\ell,j+1}|$ at the same jump point j, meaning that they are utilizing the same coset state tuple ([LLQZ22]) or two quantum money banknotes must come from the same initial banknote ([AC12]); where they randomly guess this critical index and place the $1 \to 2$ challenge there. However, the problem in our case is that the possible jump points j_ℓ are in $\{0, 1, \dots, 2^\lambda - 1\}$, whereas we only have $k+1$ freeloaders. This creates a multitude of problems: (i) we cannot conclude that there will be a non-negligible jump since the average step between $q_{\ell,0}$ and $q_{\ell,2^\lambda}$ is $\gamma/2^\lambda$, which is negligible, (ii) even if there is a non-negligible jump, we cannot apply the pigeonhole principle to guarantee that there is a pair of freeloaders ℓ, ℓ' that have the jump index $j_\ell = j_{\ell'}$ since we have 2^λ slots for $k+1$ freeloaders. Further, note that even if both of the previous concerns worked out and the two freeloaders' non-negligible jump indices coincide, we cannot actually test the freeloaders with respect to all $\mathcal{D}_j$ to find it or randomly guess it since there are exponentially many possibilities. However, a careful reader might guess that thanks to the IBE security, the challenge ciphertext distributions above actually *collapse* around k points: $j = id_1, \dots, id_k$, the identity strings of the secret keys obtained by the adversary. That is, we claim that jumps can only happen at indices j that correspond to some id_i. The reason is that, informally, the difference between $\mathcal{D}_j$ and $\mathcal{D}_{j+1}$ only occurs when the obfuscated ciphertext program is evaluated at $id = j$, in which case the output is IBE encryptions of m and $\top$ respectively, both under the identity j. However, if j is not one of id_i, then the different outputs of these programs will be IBE ciphertexts that are indistinguishable to the adversary, by the security of IBE. Therefore, no freeloader can detect this change, and there cannot be a jump between $q_{\ell,j}$ and $q_{\ell,j+1}$. This (i) allows us to conclude that for each freeloader there must be a γ/k jump (which is non-negligible) at one of $j = id_1, \dots, id_k$, and (ii) since the jump points are now all in $\{id_1, \dots, id_k\}$, we can apply a pigeonhole argument to say that there is two freeloaders have the same jump point since there are $k+1$ freeloaders with k jump slots. However, note that the ciphertext programs are only $i\mathcal{O}$ programs and not ideal oracles, therefore, the above argument is only informal and needs to be proven. Overall, while the above intuitions are the crux of our technique, formalizing these requires care and the full proof delicately intertwines all these observations, whilst also dealing with further technical problems. We refer the reader to Section 7.3 for the full proof. We also need some new results on *collusion-resistant MoE for pseudorandom coset states*, which we prove in Section 5.

[20] Here, we are talking about any freeloader program/state, not necessarily the initial ones, since the state of the freeloaders has changed since we already applied the previous tests $\mathsf{PI}_{\mathcal{D}_j}$ for $j = 0, \dots, 2^\lambda - 1$

2.3 Public-Key Functional Encryption with Copy-Protected Functional Keys

In the setting of functional encryption, we now have functional keys, where a functional key for a function f allows one to obtain $f(m)$ given the encryption $\mathsf{Enc}(m)$, and nothing else. Similar to PKE (Section 2.1), for functional encryption with copy-protected keys, we require that an adversary that obtains k copy-protected (functional) keys cannot create $k+1$ working keys (for any functions). We also allow the adversary to obtain classical functional keys. See Section 8.1 for details of our model. We move onto our construction. The starting point is our public-key encryption scheme. To generate a quantum copy-protected key for a function f, we sample a random id as before, but now we generate the coset tuple using the randomness $F(K, id||f)$ rather than $F(K, id)$. Basically, the coset states are now associated with both the function f and a random id. We note that the random identity is still required, since we allow the adversary to query for multiple copy-protected keys for the same function f. We also change our ciphertexts so that they now output an IBE encryption of $f(m)$ under the identity $id||f$, rather than outputting an encryption of m. We refer the reader to Section 8.2 for the full construction.

Proving Security: Building and Using Puncturable Functional Encryption

Proof of security for our FE scheme will be similar to the proof of our PKE scheme (Section 2.2). In particular, we will now identify identity string-function pairs (associated with the functional keys) with elements of $\{0, 1, \ldots, 2^\lambda \cdot 2^\lambda\}$, and have ciphertext distributions $\mathcal{D}_i$ for all such elements. While we have $2^{2\lambda}$ jump points, similar to PKE, we can argue that they can occur only at k points: $j = id_1||f_1, \ldots, id_k||f_k$, where $f_1, \ldots, f_k$ are the functions the pirate adversary has queried in copy-protected mode and $id_1, \ldots, id_k$ are the associated identity strings in the same order. The reason is that, for other values, either (i) the adversary will not have the IBE secret key for the identity $id||g$ (meaning that it has not queried for the function g), or (ii) we will have $g(m_0) = g(m_1)$ (if it has queried for the function g in the classical mode). Thus, $\mathcal{D}_i$ and $\mathcal{D}_{i+1}$ will be indistinguishable at points other than ones listed above; either by IBE security or since the PCt will output $g(m_0) = g(m_1)$ in both distributions.

There is one caveat left. As discussed before, in our copy-protection security proofs, we crucially rely on *projective implementations* [Zha20] to estimate the success of the freeloader adversaries for the task where they are given an encryption of m^b with random $b \leftarrow \{0, 1\}$ and they output a prediction b' for it. This allows us to simultaneously extract vectors from two entangled freeloader adversaries. While projective implementations are in general inefficient, [Zha20] also gives an efficient algorithm (called *approximated projective implementation*) that approximates it well, using a technique similar to the celebrated witness-preserving QMA amplification result of [MW04]. Crucially, we note that above decryption process between the challenger and freeloader, for which we estimate the success probability, is non-interactive (i.e., not single round). However, in a copy-protected functional encryption security game, the freeloader adversaries will be allowed to query for more functional keys after they receive their challenge ciphertexts, for any polynomial number of rounds. Therefore, we will not able to use the approximated projective implementation as-is to estimate the success probability of a freeloader adversary for functional encryption. While one solution might be to try and generalize approximate projective implementations to interactive procedures, given that the original technique of [MW04] also only applies to QMA (which is single round), this might be a challenging task.

We side-step the issue above using a classical solution. We define a variation of our scheme where the challenger gives the freeloader adversaries a *punctured master secret key pmsk* along with their challange ciphertext. This punctured key has the challenge messages m^0, m^1 chosen by the adversary hardcoded, and it takes in a function f and outputs the secret key for f if $f(m^0) = f(m^1)$. Then, since the freeloader adversaries can simulate (using this punctured key $pmsk$) themselves any key queries that they want to make after seeing the challenge ciphertext, we remove the interaction between the freeloaders and the challenger. As a result, we are again able to use approximate projective implementations in our technique.

The only remaining challenge is making sure that our functional encryption construction is still secure when the adversaries obtain this punctured master secret key, which is an obfuscated program that contains the master secret key msk. While $pmsk$ will only answer the queries on functions that the adversary was allowed to query for anyways, the problem is that we are using indistinguishability obfuscation rather than black-box obfuscation to compute $psmk$. To resolve this issue, we upgrade our FE construction to use an identity-based encryption scheme with puncturable master secret keys (Section 6). In such a scheme, we are able to produce a master secret key that can issue identity keys for any identity other than the identity it was punctured at. When we are proving the security of our functional encryption scheme, we will construct hybrids corresponding to all possible $id||f$. Moving between each hybrid, we only need to rely on the security of IBE at this identity. Therefore, in our security proof, we will not only use a puncturing argument inside our obfuscated ciphertext program PCt, but we will also puncture the IBE master secret key inside $pmsk$ at $id||f$. Thus, we will be able to rely on the security of IBE even when the adversary has $pmsk$. We refer the reader to Section 8.3 for the full proof.

2.4 PRFs and Signature Schemes with Copy-Protected Secret Keys

Let us first describe the setting. In the case of PRFs, we imagine a quantum key generation algorithm that, given the PRF key K, can generate copy-protected keys that can be used to evaluate the PRF $F(K, \cdot)$ any number of times. For copy-protection, we require that given k such keys, the adversary cannot create $k + 1$ freeloaders that can distinguish $F(K, x)$ versus a random string from the co-domain of F, given uniformly at random x.[21] See Section 10 for more details. In the case of signatures, we have copy-protected re-usable signing keys that can sign any message. Similar to above, given k such keys, pirate outputs $k + 1$ freeloaders, and we ask the freeloaders to sign random messages.[22] See Section 10 and Section 9 for more details.

Our signature scheme will be the same as our PRF scheme, where the signature on m will be the PRF evaluation $F(K, m)$, with the difference from the PRF scheme being that we will also have a verification key. Similar to the signature scheme construction of Sahai and Waters [SW14], the verification key will be an obfuscated program that verifies a message-signature pair (m, σ) by checking $f(\sigma) = f(F(K, m))$ where f is a one-way function. Due to these similarities, we only discuss our signature scheme here.

In our signature scheme, a copy-protected signing key will consist of two parts: (i) a coset state tuple generated using the randomness $F(K'', id)$ for random id, similar to our PKE scheme; (ii) an obfuscated signing program PSign_K. The program PSign_K will take as input a message m, along with id and vectors $(v_i)_{i\in[c]}$. Similar to the ciphertext programs PCt in our PKE construction, PSign_K will verify that the vectors $(v_i)_{i\in[c]}$ are in the correct cosets $A_i + s_i$ or $A_i^{\perp} + s'_i$, depending on the i-th bit of m, where the tuple (A_i, s_i, s'_i) is associated with id. Informally, since the challenge messages m^1, m^2 are random, similar to r^1, r^2 in the PKE case, we will be able to violate the

[21] Note that x being randomized and being revealated after the splitting is required, since otherwise the pirate can evaluate the PRF before splitting into freeloaders, and it can simply give the classical evaluation results to the freeloaders.

[22] As in the case of PRFs, known/deterministic messages can be signed before the split by the pirate, hence, random challenge messages are required.

monogamy-of-entanglement game, given freeloader adversaries that can sign these message - arriving at a contradiction. However, since we are using $i\mathcal{O}$ and not black-box obfuscation to obfuscate PSign_K, some information about K might leaking, allowing the adversary to sign messages without querying the program with correct vectors. Similar to [CLLZ21, LLQZ22], we use the *hidden trigger technique* of [SW14] to solve this issue and reduce the security of our signature scheme to that of our copy-protected PKE scheme.

Hidden triggers, introduced by [SW14] to construct deniable encryption, is a sparse set of inputs that can be efficiently sampled and are pseudorandom, even given a program that *uses* these inputs. In the case of [CLLZ21, LLQZ22], their set of hidden trigger inputs are special encodings of the ciphertexts of their copy-protected PKE scheme. Using this technique, they embed a separate thread in PSign_K that detects if the message m is a trigger input, and in that case, executes the embedded ciphertext program PCt (which is a PKE encryption of $F(K, m)$) in this input instead of normal execution. This allows them to reduce the task of finding the signature $F(K, m)$ for a message m to the task of decrypting a PKE ciphertext encrypting $F(K, m)$ (hence reducing security to their PKE scheme), by undetectably replacing the random challenge messages to be signed with hidden triggers.

In our case, two new issues arise. First, as discussed, to achieve collusion-resistance, our PKE ciphertext programs crucially output IBE ciphertexts upon successful coset vector verification, meaning that they are randomized programs, which makes it more challenging to encode them as hidden triggers. We solve this issue as follows. Inside the ciphertext program, we expand the hidden signature $F(K, m)$ using a PRG (since there are various length requirements on the input-output size to be able to use the hidden trigger technique), and we use part of the expanded string as a PRF key to supply randomness to $\mathsf{IBE.Encrypt}$.

Secondly, the previous work [CLLZ21, LLQZ22] crucially rely on puncturing the PRF key K at all the challenge points $m_1, \ldots, m_{k+1}$, to replace these challenge messages with hidden trigger inputs and utilize the hidden thread in PSign. However, in our case, we would have to puncture PRF key at $k+1$ points since we have $k+1$ freeloaders/challenges, where k is not a-priori bounded - this is not possible since the sizes of the punctured key and the obfuscated programs would need to grow with k. We solve this issue by making our hidden trigger inputs *publicly generatable*, that is, by arguing that hidden trigger inputs are indistinguishable from uniform strings even given a program that generates these inputs (which needs to include the PRF key K). This allows us to only prove that a single challenge message is indistinguishable from a single hidden trigger input, and then we simply rely on the hybrid lemma to conclude the same result for any number of challenge messages, since the trigger inputs can now be generated by the adversary itself during the hybrid lemma argument. We use a new *prefix-puncturing* (Definition 4) argument for the PRF key K to achieve publicly-generatable hidden triggers for our scheme. The full proof is technical, we refer the reader to Section 9.5 and Appendix D for the full proofs. We believe our technique might be of independent interest and might find applications in classical cryptography.

2.5 Impossibility of Hyperefficient Shadow Tomography

As discussed in the introduction, an important corollary of our result is the impossibility of hyperefficient shadow tomography. Suppose a shadow tomography procedure exists. We now describe a generic attack on copy-protection, given by [Aar18] and adapted to the case of copy-protecting decryption keys by [SW22], that uses shadow tomography. Let $s(\lambda)$ be the size of the ciphertexts of a public-key encryption scheme PKE with collusion-resistant copy-protected secret keys, for 1-bit messages. Then, define the set of measurements $\{E_{ct}\}_{ct \in \{0,1\}^{s(\lambda)}}$ as follows: E_{ct} is the binary measurement $\mathsf{PKE.Dec}(\cdot, ct)$. That is, given a state (which will be a copy-protected secret key in our case), E_{ct} is the binary measurement implemented by running $\mathsf{PKE.Dec}$ on ρ and accepting if it outputs 1. Then, it is easy to see that once we obtain the estimates of the acceptance probabilities of all E_{ct} for the state ρ where ρ is the copy-protected secret key, when we are given a ciphertext ct, we can simply use this estimate to tell if ct is an encryption of 1 or 0, since E_{ct} would *accept* ρ if ct is an encryption of 1, and reject it otherwise. Since these estimates are classical values, given some number of keys we can perform shadow tomography and then we can create any number of decryption programs.

The attack above is used by [Aar18, SW22] to conclude that *unconditional* collusion-resistant copy-protection is impossible, since [Aar18] gives a shadow tomography procedure that uses polynomially many copies of a state ρ, however, the procedure takes exponential time. Now, the question is, does there exist a *hyperefficient* shadow tomography procedure? We observe that the measurement set $\{E_{ct}\}_{ct \in \{0,1\}^{s(\lambda)}}$ above is actually implemented by a uniform algorithm: $\mathsf{PKE.Dec}(\cdot, \cdot)$. Hence, if there exists a hyperefficient shadow tomography (Definition 49) procedure, it would output (a classical description of) a quantum circuit that can estimate all E_{ct}, given time and number of copies that are both $\mathsf{poly}\big(|\rho|, \log\big(2^{s(\lambda)}\big)\big) = \mathsf{poly}(\lambda)$. However, this would break the security of our collusion resistant copy-protected PKE scheme (Section 7), since we can query for sufficiently many keys, perform the shadow tomography and freely distribute the resulting classical information. Thus, we conclude that hyperefficient shadow tomography is not possible. We refer the reader to Section 12 for more details.

2.6 Related Works

Copy-Protection See Section 1 for an overview of work on copy-protecting general functionalities [Aar09, ALL$^+$21], secret keys of a PKE scheme [GZ20, CLLZ21, LLQZ22], PRF keys [CLLZ21, LLQZ22], and signing keys [LLQZ22]. Kitagawa and Nishimaki [KN22] defined functional encryption with copy-protected functional keys in the weaker $\mathsf{1\text{-}copy} \rightarrow \mathsf{2\text{-}copy}$ model where the adversary can only obtain one copy-protected functional key and the freeloaders cannot query for more functional keys after receiving their challenge ciphertexts. They showed how to construct secure schemes in this model from any public-key encryption scheme with copy-protected secret keys, using $i\mathcal{O}$. Coladangelo, Majenz, and Poremba [CMP20] and Ananth et al. [AKL$^+$22] showed how to construct copy-protection for point functions and compute-and-compare functions in the quantum random oracle model.

Secure Leasing Ananth and La Placa [ALP21] introduced *secure software leasing*, which is a weaker version of copy-protection where the adversaries are only prevented from creating two copies of their program that can both be run using the *honest* evaluation algorithm. [ALP21] also show that even this weaker notion is impossible to achieve for all unlearnable programs *in the plain model*[23], based on some standard assumptions. Various work also define a variant where we require that the adversary cannot produce at the same time a working copy (now allowed to be run with any algorithm) and a valid *deletion certificate* for a program that passes the verification of the software vendor. Note that both of these variants are implied by copy-protection [KN22, ÇGLZR24]: we can always let the deletion certificate to be the copy-protected program itself and the deletion certificate verification procedure can simply test the returned program on various inputs. Various works [ALP21, ALL$^+$21, KNY21, KN22, BGG$^+$23] construct secure leasing schemes for various primitives such as functional encryption, PRFs, indistinguishability obfuscation, based on various assumptions. As discussed above, since our schemes are unclonable, they also give publicly verifiable

[23] Note that the results of [Aar09] and [ALL$^+$21] do not contradict this since they are in the oracle model.

securely leasable schemes for the same primitives, such as functional encryption.

Functional Encryption [CGJS15] construct delegatable functional encryption from hierarchical identity-based encryption (HIBE) and indistinguishability obfuscation where the ciphertext is an obfuscated program that outputs a HIBE ciphertext, similar to our FE construction. Bitansky and Vaikuntanathan [BV18], Kitagawa et al. [KNT22] and Yang et al. [YAL$^+$19] construct what they call *puncturable functional encryption*, however, their definitions are completely different from ours (and each other) and are incomparable to our model. In the first two, they construct symmetric-key functional encryption whose secret keys can be punctured at a message or a tag. The goal is to construct indistinguishability obfuscation and succinctness is an important property for their functional encryption schemes. In [YAL$^+$19], they construct a scheme where a functional key can be punctured at a ciphertext. Different from both works, in our classical functional encryption scheme with puncturable master key (Section 8), we will have a *public-key* scheme whose master secret key can be punctured at all functions that are not *differentiating* m_0, m_1.

2.7 Organization

In Section 4, we recall some technical results and also show some new ones for projective implementations, which are needed in our copy-protection security proofs.

In Section 5, we show some new collusion-resistant monogamy-of-entanglement results for coset states, which are again needed in our proofs.

In Section 6, we show how to construct a puncturable identity-based encryption scheme (which is needed for our copy-protected FE scheme) and a puncturable functional encryption scheme (which uses techniques similar to our copy-protected FE scheme and might be of independent interest).

In Section 7, Section 8, Section 9, Section 10 we give our copy-protected public-key encryption, public-key functional encryption, signature and PRF schemes with collusion-resistant copy-protected keys, respectively, along with their security proofs.

In Section 11, we give collusion-resistant copy-protection schemes for all unlearnable functionalities.

In Section 12, we show the impossibility of hyperefficient shadow tomography.

3 Preliminaries

3.1 Notation

All of our assumptions (e.g. existence of one-way functions) will be implicitly post-quantum.

We write λ to denote the security parameter. We write $\mathsf{poly}(\cdot)$ to denote a polynomial function. We write $f(\lambda) \leq \mathsf{negl}(\lambda)$ or $f(\lambda) < \mathsf{negl}(\lambda)$ and say that $f(\cdot)$ is negligible if for any polynomial $p(\cdot)$, there exits λ_0 such that $f(\lambda) < \frac{1}{p(\lambda)}$ for all $\lambda > \lambda_0$. We will write $\mathsf{subexp}(\cdot)$ to mean a subexponential function, meaning, $f(n) = 2^{n^c}$ for some constant $0 < c < 1$ and all sufficiently large n.

We say that an algorithm is efficient if it is quantum polynomial time (QPT), that is, there exists a uniform family of polynomial size quantum circuits that computes it. Unless otherwise stated, we will consider non-uniform QPT adversaries. We use the term *subexponentially secure* to mean either that the advantage of any *QPT* or *subexponential* time adversary is $\mathsf{subexp}(-\lambda)$, the distinction will be clear from context. In our constructions, we will rely on the subexponential security of the underlying primitives for specific subexponential functions, such as $2^{-\lambda^c}$-security. However (unless otherwise specified) this is equivalent to assuming subexponential security for any subexponential function, since we can scale the security parameter by a polynomial.

We write $|X - Y|$ to denote the total variation distance between two classical random variables and we write $\|\rho - \sigma\|_{Tr}$ to denote the trace distance between two quantum random variables (i.e. density matrices) ρ, σ. For a sequence of (classical or quantum) random variables $X = \{X_\lambda\}_\lambda, Y = \{Y_\lambda\}_\lambda$, we write $X \approx_\varepsilon Y$ to mean $|X - Y| < \varepsilon$ or $\|X - Y\|_{Tr} < \varepsilon(\lambda)$; and we write $X \approx^c_\varepsilon Y$ to mean $\left|\Pr\left[\mathcal{A}(1^\lambda, X) = 1\right] - \Pr\left[\mathcal{A}(1^\lambda, Y) = 1\right]\right| < \varepsilon(\lambda)$ for any appropriately (will be clear from context) bounded (i.e computational) adversary $\mathcal{A}$. Both are only for all sufficiently large λ. In both cases we omit ε when $\varepsilon = \mathsf{negl}(\lambda)$ and we will omit specifying the adversarial constraint when the constraint is that the adversary runs in polynomial time.

We will write $\mathcal{M}$ to denote a message space (e.g., $\{0,1\}^{m(\lambda)}$).

For a string x, we will write $(x)_i$ to denote the i-th character/bit.

We assume that the reader is familiar with the basics of quantum information theory. We will use the quantum register model, where a register is an object that has a quantum state that evolves when we act on it. We will usually write R to denote a quantum register and $\mathcal{H}$ to denote a Hilbert space. We refer the reader to [NC10] and [Wat18] for a comprehensive review of quantum information theory.

Wherever we use indistinguishability obfuscation $i\mathcal{O}$, we assume that the obfuscated circuits are appropriately padded.

3.2 Puncturable Pseudorandom Functions

In this section, we introduce puncturable pseudorandom functions.

Definition 1 ([SW14])**.** *A puncturable pseudorandom function (PRF) is a family of functions $\{F : \{0,1\}^{c(\lambda)} \times \{0,1\}^{m(\lambda)} \to \{0,1\}^{n(\lambda)}\}_{\lambda \in \mathbb{N}^+}$ with the following efficient algorithms.*

- $F.\mathsf{Setup}(1^\lambda)$: *Takes in a security parameter and outputs a key in $\{0,1\}^{c(\lambda)}$.*
- $F(K, x)$:[24] *Takes in a key and an input, outputs an evaluation of the PRF.*
- $F.\mathsf{Puncture}(K, S)$: *Takes as input a key and a set $S \subseteq \{0,1\}^{m(\lambda)}$, outputs a punctured key.*

We require the following.

Correctness. *For all efficient distributions $\mathcal{D}(1^\lambda)$ over the power set $2^{\{0,1\}^{m(\lambda)}}$, we require*

$$\Pr\left[\forall x \notin S \quad F(K_S, x) = F(K, x) : \begin{array}{c} S \leftarrow \mathcal{D}(1^\lambda) \\ K \leftarrow \mathsf{KeyGen}(1^\lambda) \\ K_S \leftarrow \mathsf{Puncture}(K, S) \end{array}\right] = 1.$$

Puncturing Security *We require that any stateful QPT adversary $\mathcal{A}$ wins the following game with probability at most $1/2 + \mathsf{negl}(\lambda)$.*

1. *$\mathcal{A}$ outputs a set S.*
2. *The challenger samples $K \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $K_S \leftarrow \mathsf{Puncture}(K, S)$*
3. *The challenger samples $b \leftarrow \{0,1\}$. If $b = 0$, the challenger submits $K_S, \{F(K, x)\}_{x \in S}$ to the adversary. Otherwise, it submits $K_S, \{y_s\}_{s \in S}$ to the adversary where $y_s \leftarrow \{0,1\}^{n(\lambda)}$ for all $s \in S$.*

[24] We overload the notation and write F to both denote the function itself and the evaluation algorithm.

4. The adversary outputs a guess b' and we say that the adversary has won if $b' = b$.

Definition 2 (Injective PRF [SW14])**.** *A PRF family F is said to be* statistically injective *with failure probability $\varepsilon(\lambda)$ if, with probability $1 - \varepsilon(\lambda)$ over the sampling of the key K, the function $F(K, \cdot)$ is injective.*

Definition 3 (Extracting PRF [SW14])**.** *A PRF family F with input space $\{0,1\}^{m(\lambda)}$ and output space $\{0,1\}^{n(\lambda)}$ is said to be extracting with error $\varepsilon(\lambda)$ for min-entropy $k(\lambda)$ if for any distribution of X over $\{0,1\}^{m(\lambda)}$, we have $(K, F(K,X)) \approx_{\varepsilon(\lambda)} (K, U)$ where $K \leftarrow F.\mathsf{Setup}(1^\lambda)$ and U is sampled uniformly at random from $\{0,1\}^{n(\lambda)}$.*

Theorem 6 ([SW14, GGM86, Zha12a])**.** *Let $n(\cdot), m(\cdot), e(\lambda), k(\lambda)$ be efficiently computable functions.*

- *If (post-quantum) one-way functions exist, then there exists a (post-quantum) puncturable PRF with input space $\{0,1\}^{m(\lambda)}$ and output space $\{0,1\}^{n(\lambda)}$.*
- *If we assume subexponentially-secure (post-quantum) one-way functions exist, then for any $c > 0$, there exists a (post-quantum) $2^{-\lambda^c}$-secure[25] puncturable PRF against subexponential time adversaries with input space $\{0,1\}^{m(\lambda)}$ and output space $\{0,1\}^{n(\lambda)}$.*
- *If (post-quantum) one-way functions exist, then there exists a puncturable extracting PRF with error $2^{-e(\lambda)}$ for min-entropy $k(\lambda)$, with input space $\{0,1\}^{m(\lambda)}$ and output space $\{0,1\}^{n(\lambda)}$, if $m(\lambda) \geq k(\lambda) \geq n(\lambda) + 2 \cdot e(\lambda) + 2$. The same result follows for the subexponential case as above.*
- *If (post-quantum) one-way functions exist, then there exists a puncturable statistically injective PRF with error $2^{-e(\lambda)}$, with input space $\{0,1\}^{m(\lambda)}$ and output space $\{0,1\}^{n(\lambda)}$, if $n(\lambda) \geq 2 \cdot m(\lambda) + e(\lambda)$. The same result follows for the subexponential case as above.*

We also introduce PRFs with prefix puncturing, similar to puncturable PRFs and prefix constrained PRF keys [BW13][26].

Definition 4. *A prefix puncturable pseudorandom function (PRF) is a PRF $\{F : \{0,1\}^{c(\lambda)} \times \{0,1\}^{m(\lambda)} \to \{0,1\}^{n(\lambda)}\}_{\lambda \in \mathbb{N}^+}$ with the following additional algorithm.*

- *$F.\mathsf{Puncture}(K, pre)$: Takes as input a key and a prefix pre of length at most $m(\lambda)$, outputs a punctured key.*

We require the following.

Correctness. *For all efficient distributions $\mathcal{D}(1^\lambda)$ over the set $\bigcup_{\ell \leq m(\lambda)} \{0,1\}^\ell \otimes \{0,1\}^{m(\lambda)}$, we require*

$$\Pr\left[pre \text{ is a prefix of } x \bigvee F(K\{pre||\cdot\}, x) = F(K, x) : \begin{array}{c} (pre, x) \leftarrow \mathcal{D}(1^\lambda) \\ K \leftarrow \mathsf{KeyGen}(1^\lambda) \\ K\{pre||\cdot\} \leftarrow \mathsf{Puncture}(K, pre) \end{array} \right] = 1.$$

[25] While the original results are for negligible security against polynomial time adversaries, it is easy to see that they carry over to subexponential security. Further, by scaling the security parameter by a polynomial and simple input/output conversions, subexponentially secure (for any exponent c') one-way functions is sufficient to construct for any c a puncturable PRF that is $2^{-\lambda^c}$-secure.

[26] In a prefix constrained PRF key, one requires that given the constrained key, any input x that starts with the prefix can be evaluated, and all other PRF output values remain pseudorandom. In our setting we will require the opposite: for any input that starts with the prefix, the output will remain pseudorandom, while other inputs can be evaluated using the punctured key.

Puncturing Security *We require that any stateful QPT adversary $\mathcal{A}$ wins the following game with probability at most $1/2 + \mathsf{negl}(\lambda)$.*

1. *$\mathcal{A}$ outputs a prefix pre of length at most $m(\lambda)$ and a string $x \in \{0,1\}^{m(\lambda)}$ such that pre is a prefix of x.*
2. *The challenger samples $K \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $K\{pre||\cdot\} \leftarrow \mathsf{Puncture}(K, pre)$.*
3. *The challenger samples $b \leftarrow \{0,1\}$. If $b = 0$, the challenger submits $K\{pre||\cdot\}, F(K,x)$ to the adversary. Otherwise, it submits $K\{pre||\cdot\}, y$ to the adversary where $y \leftarrow \{0,1\}^{n(\lambda)}$.*
4. *The adversary outputs a guess b' and we say that the adversary has won if $b' = b$.*

Theorem 7. *Let $n(\cdot), m(\cdot), e(\lambda), k(\lambda)$ be efficiently computable functions. If (post-quantum) one-way functions exist, then there exists a prefix puncturable extracting PRF with error $2^{-e(\lambda)}$ for min-entropy $k(\lambda)$, with input space $\{0,1\}^{m(\lambda)}$ and output space $\{0,1\}^{n(\lambda)}$, if $m(\lambda) \geq k(\lambda) \geq n(\lambda) + 2 \cdot e(\lambda) + 2$. The same result follows for the subexponential case as above.*

The above theorem follows in two steps. First, we can obtain a prefix puncturable PRF using the GGM construction [GGM86] (which is post-quantum secure [Zha12a]): we partially open the evaluation tree on the key K according to $pre = b_1 \dots b_\ell$, and then output the keys for leaves $\overline{b_1}, b_1\overline{b_2}, b_1 b_2 \overline{b_3}, \dots, b_1 \cdots b_{\ell-1}\overline{b_\ell}$. Then, by an application of [SW14, Theorem 3] we can make it extracting.

3.3 Indistinguishability Obfuscation

In this section, we introduce indistinguishability obfuscation.

Definition 5. *An indistinguishability obfuscation scheme $i\mathcal{O}$ for a class of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ satisfies the following.*

Correctness. *For all $\lambda, C \in \mathcal{C}_\lambda$ and inputs x, $\Pr\left[\tilde{C}(x) = C(x) : \tilde{C} \leftarrow i\mathcal{O}(1^\lambda, C)\right] = 1$.*

Security. Let $\mathcal{B}$ be any QPT algorithm that outputs two circuits $C_0, C_1 \in \mathcal{C}$ of the same size, along with auxiliary information, such that $\Pr\left[\forall x\ C_0(x) = C_1(x) : (C_0, C_1, \mathsf{R}_{\mathsf{aux}}) \leftarrow \mathcal{B}(1^\lambda)\right] \geq 1 - \mathsf{negl}(\lambda)$. Then, for any QPT adversary $\mathcal{A}$,

$$\Big| \Pr\left[\mathcal{A}(i\mathcal{O}(1^\lambda, C_0), \mathsf{R}_{\mathsf{aux}}) = 1 : (C_0, C_1, \mathsf{R}_{\mathsf{aux}}) \leftarrow \mathcal{B}(1^\lambda)\right] - \Pr\left[\mathcal{A}(i\mathcal{O}(1^\lambda, C_1), \mathsf{R}_{\mathsf{aux}}) = 1 : (C_0, C_1, \mathsf{R}_{\mathsf{aux}}) \leftarrow \mathcal{B}(1^\lambda)\right] \Big| \leq \mathsf{negl}(\lambda).$$

3.4 Functional Encryption

In this section we introduce the basic definitions of functional encryption schemes.

Definition 6 (Functional encryption). *A functional encryption scheme for a class of functions $\mathfrak{F}$ consists of the following algorithms that satisfy the correctness and security guarantees below.*

- $\mathsf{Setup}(1^\lambda)$*: Outputs a master secret key msk and a public key pk.*

- $\mathsf{KeyGen}(msk, f)$: *Takes in the master secret key and a function $f \in \mathfrak{F}$, outputs a functional key for f.*
- $\mathsf{Enc}(pk, m)$: *Takes in the public key and a message m, outputs an encryption of m.*
- $\mathsf{Dec}(sk, ct)$: *Takes in a functional key sk and a ciphertext, outputs a message or $\perp$.*

Correctness *For all functions $f \in \mathfrak{F}$ and all messages m, we require the following.*

$$\Pr\left[\mathsf{Dec}(sk, ct) = f(m) : \begin{array}{c} msk, pk \leftarrow \mathsf{Setup}(1) \\ sk \leftarrow \mathsf{KeyGen}(msk, f) \\ ct \leftarrow \mathsf{Enc}(pk, m) \end{array}\right] = 1.$$

Adaptive indistinguishability security *Consider the following game between a challenger and an adversary $\mathcal{A}$.*

$\underline{\mathsf{FE} - \mathsf{IND}(\lambda, \mathcal{A})}$

1. *Challenger samples the keys $msk, pk \leftarrow \mathsf{Setup}(1)$.*
2. *The adversary receives pk. It makes polynomially many queries by sending functions $f \in \mathcal{F}$ and receiving the corresponding functional key $sk_f \leftarrow \mathsf{KeyGen}(msk, f)$.*
3. *The adversary outputs challenge messages m_0, m_1.*
4. *The challenger samples a challenge bit $b \leftarrow \{0, 1\}$ and prepares $ct \leftarrow \mathsf{Enc}(pk, m_b)$.*
5. *The adversary receives ct, and it makes polynomially many functional key queries.*
6. *The adversary outputs a guess b'.*
7. *The challenger checks if $f(m_0) = f(m_1)$ for all f queried by the adversary. If not, it outputs 0 and terminates.*
8. *The challenger outputs 1 if $b' = b$.*

We require that for any QPT adversary $\mathcal{A}$,

$$\Pr[\mathsf{FE} - \mathsf{IND}(\lambda, \mathcal{A}) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

If the adversary outputs the challenge messages before the keys are sampled, we call it selective indistinguishability security.

3.5 Quantum Information Theory

In this section, we present various technical lemmas regarding quantum information theory.

Lemma 1 (Almost As Good As New Lemma [Aar16], verbatim)**.** *Let ρ be a mixed state acting on $\mathbb{C}^d$. Let U be a unitary and $(\Pi_0, \Pi_1 = I - \Pi_0)$ be projectors all acting on $\mathbb{C}^d \otimes \mathbb{C}^{d'}$. We interpret (U, Π_0, Π_1) as a measurement performed by appending an ancillary system of dimension d' in the state $|0\rangle\langle 0|$, applying U and then performing the projective measurement Π_0, Π_1 on the larger system. Assuming that the outcome corresponding to Π_0 has probability $1 - \varepsilon$, we have*

$$\left\|\rho - \rho'\right\|_{Tr} \leq \sqrt{\varepsilon}$$

where ρ' is the state after performing the measurement, undoing the unitary U and tracing out the ancillary system.

We sometimes also use the following related result.

Lemma 2 (Gentle Measurement Lemma [Wil15])**.** *Let E be a POVM element and ρ be a state of appropriate dimension. Suppose the outcome E has a high probability of occuring, that is, $\mathrm{Tr}\{E\rho\} \geq 1-\varepsilon$. Then, if we apply a canonical implementation, $\sqrt{E}$, of this measurement, the post-measurement state conditioned on this outcome is close to the original state:*

$$\left\| \rho - \frac{\sqrt{E}\rho\sqrt{E}}{\mathrm{Tr}\{E\rho\}} \right\|_{Tr} \leq \sqrt{\varepsilon}.$$

Theorem 8 (Implementation Independence of Measurements on Bipartite States)**.** *Let $\Lambda = \{M_i\}_{i\in\mathcal{I}}, \Lambda' = \{E_i\}_{i\in\mathcal{I}}$ be two general measurements whose POVMs are equivalent, that is, $M_i^\dagger M_i = E_i^\dagger E_i$ for all $i \in \mathcal{I}$.*

Let ρ be any bipartite state whose first register has the appropriate dimension for Λ, Λ'. Then, the post-measurement state of the second register conditioned on any outcome $i \in \mathcal{I}$ is the same when either Λ or Λ' is applied to the first register of ρ. That is,

$$(\mathrm{Tr} \otimes I) \frac{(M_i \otimes I)\rho(M_i^\dagger \otimes I)}{\mathrm{Tr}\Big\{(M_i \otimes I)\rho(M_i^\dagger \otimes I)\Big\}} = (\mathrm{Tr} \otimes I) \frac{(E_i \otimes I)\rho(E_i^\dagger \otimes I)}{\mathrm{Tr}\Big\{(E_i \otimes I)\rho(E_i^\dagger \otimes I)\Big\}}$$

Proof. See Appendix A.1. □

Lemma 3 (Trace Distance Conditioned on Measurement Outcome)**.** *Let $M = \{M_i\}_{i\in\mathcal{I}}$ be a general measurement and ρ, σ be two states of appropriate dimension such that $\|\rho - \sigma\|_{Tr} \leq \varepsilon$. Let p_i denote probability of outcome i when M is applied to ρ, that is $p_i = \mathrm{Tr}\{M_i\rho\}$. Then,*

$$\|\rho' - \sigma'\|_{Tr} \leq \frac{3\varepsilon}{2p_i}$$

where ρ', σ' are post-measurement states conditioned on outcome i when the measurement M is applied to ρ, σ, respectively. That is, $\rho' = \frac{M_i\rho M_i^\dagger}{\mathrm{Tr}\left\{M_i\rho M_i^\dagger\right\}}$ and $\sigma' = \frac{M_i\sigma M_i^\dagger}{\mathrm{Tr}\left\{M_i\sigma M_i^\dagger\right\}}$.

Proof. See Appendix A.2. □

Theorem 9. *Let ρ be a bipartite state and $\Lambda = \{\Pi_1, \dots\}, \Lambda' = \{\Pi_1', \dots\}$ be two projective measurements over each of these registers, respectively. Suppose*

$$\mathrm{Tr}\big\{\Pi_1 \otimes \Pi_1'\rho\big\} \geq 1 - \varepsilon.$$

Let $M = \{M_i\}_{i\in\mathcal{I}}$ be a general measurement over the first register and fix any $i \in \mathcal{I}$. Let τ denote the post-measurement state of the second register after applying the measurement M on the first register of ρ and conditioned on obtaining outcome i. Let p_i denote probability of outcome i, that is $p_i = \mathrm{Tr}\Big\{(M_i \otimes I)\rho(M_i^\dagger \otimes I)\Big\}$. Then,

$$\mathrm{Tr}\big\{\Pi_1'\tau\big\} \geq 1 - \frac{3\sqrt{\varepsilon}}{2p_i}.$$

Proof. See Appendix A.3. □

Theorem 10 (Quantum Union Bound for Commuting Projectors)**.** *Let $\Pi_1, \dots, \Pi_n$ be a set of commuting projectors. Then, for any state ρ of appropriate dimension,*

$$\mathrm{Tr}[(I - \Pi_1 \dots \Pi_n)\rho] \leq \sum_{i \in [n]} \mathrm{Tr}[(I - \Pi_i)\rho].$$

Proof. While this is a folklore result, we give a proof in Appendix A.4 for completeness. □

Definition 7 (Query Algorithm)**.** *Let $\mathcal{O}$ be a function. A query algorithm $\mathcal{A}$ with oracle access to $\mathcal{O}$ is defined by an evolution unitary $U_{\mathcal{A}}$ of $\mathcal{A}$, and we also define the oracle unitary $U_{\mathcal{O}}$ as $U_{\mathcal{O}} : |w, x, y\rangle \to |w, x, \mathcal{O}(x) \oplus y\rangle$, with the registers ordered as the working register of $\mathcal{A}$, the query input register and the query output register. $\mathcal{A}$ is executed by applying $U_{\mathcal{A}}$ and then $U_{\mathcal{O}}$ in sequence, e.g., the final state of the algorithm is $(U_{\mathcal{A}} U_{\mathcal{O}})^T |\psi\rangle$ for an algorithm with initial state $|\psi\rangle$ that makes T queries. If the algorithm has classical output, the output is obtained by measuring (a part of) the working register at the end.*

Theorem 11 ([BBBV97])**.** *Let $\mathcal{A}$ be a quantum algorithm making queries to an oracle $\mathcal{O}$. Let $|\psi_t\rangle = \sum_{w,x,y} \alpha_{w,x,y} |w, x, y\rangle$ denote the joint state of the working register, the query input register and the query output register of the algorithm right before the t-th query. For a subset S of the domain of $\mathcal{O}$, let $q_S(|\psi_t\rangle) = \sum_{x \in S} |\alpha_{w,x,y}|^2$ and $q_S = \sum_t q_s(|\psi_t\rangle)$, and call q_S the query weight of S. Let $\mathcal{O}'$ be another oracle whose output differs from $\mathcal{O}$ only on points $x \in S$. Then, if $\mathcal{A}$ makes T queries to the oracle $\mathcal{O}$ and S is a subset such that $q_S \leq \varepsilon^2/T$, we have $\| |\psi\rangle\langle\psi| - |\psi'\rangle\langle\psi'| \|_{Tr} \leq \varepsilon$ where $|\psi\rangle, |\psi'\rangle$ denote the final state of the algorithm $\mathcal{A}$ when given access to the oracles $\mathcal{O}, \mathcal{O}'$ respectively.*

3.6 Compute-and-Compare Obfuscation

In this section, we introduce compute-and-compare obfuscation.

Definition 8 (Compute-and-compare program)**.** *Let $f : \{0,1\}^{a(\lambda)} \to \{0,1\}^{b(\lambda)}$ be a function, $y \in \{0,1\}^{b(\lambda)}$ be a target value and z a hidden message. The following program P, described by (f, y, z), is called a* compute-and-compare program.

$P(x)$: *Compute $f(x)$ and compare it to y. If they are equal, output z. Otherwise, output $\perp$.*

We say that a distribution $\mathcal{D}$ of such programs (along with quantum auxiliary information $\mathsf{R}_{\mathsf{aux}}$) is sub-exponentially unpredictable if for any QPT adversary, given the auxiliary information $\mathsf{R}_{\mathsf{aux}}$ and the description of f, the adversary can predict the target value y with at most subexponential probability.

Definition 9. *A compute-and-compare obfuscation scheme for a class of distributions consists of efficient algorithms* $\mathsf{CCObf.Obf}$ *and* $\mathsf{CCObf.Sim}$ *that satisfy the following. Consider any distribution $\mathcal{D}$ over compute-and-compare programs, along with quantum auxiliary input, in this class.*

Correctness. *For any function (f, y, z) in the support of $\mathcal{D}$,* $\Pr[\forall x \; D'(x) = D(x) : D' \leftarrow \mathsf{CCObf.Obf}(f, y, z)] \geq 1 - \mathsf{negl}(\lambda)$.

Security $(\mathsf{CCObf.Obf}(f, y, z), \mathsf{R}_{\mathsf{aux}}) \approx (\mathsf{CCObf.Sim}(1^\lambda, |f|, |y|, |z|), \mathsf{R}_{\mathsf{aux}})$ *where* $(f, y, z), \mathsf{R}_{\mathsf{aux}} \leftarrow \mathcal{D}(1^\lambda)$.

Theorem 12 ([CLLZ21, WZ17]). *Assuming the existence of post-quantum $i\mathcal{O}$ and LWE, there exists compute-and-compare obfuscation for any class of sub-exponentially unpredictable distributions.*

Assuming the existence of subexponentially secure $i\mathcal{O}$ and LWE against subexponential time quantum adversaries, there exists subexponentially secure compute-and-compare obfuscation against subexponential time adversaries for any class of sub-exponentially unpredictable distributions.[27]

4 Projective and Threshold Implementations

In this section, we introduce the notion of projective and threshold implementations [Zha20, ALL+21], along with their efficient versions; which are tools we use in our security proofs. Then, we recall some properties from previous work and also show some new technical results that will be needed in the security proofs of our schemes.

Definition 10 (Shift Distance [Zha20]). *Let $\mathcal{D}_0, \mathcal{D}_1$ be two distributions over $\mathbb{R}_{\geq 0}$. The shift distance with parameter $\varepsilon \geq 0$ between $\mathcal{D}_0, \mathcal{D}_1$, denoted $\Delta^{\varepsilon}_{\mathsf{Shift}}(\mathcal{D}_0, \mathcal{D}_1)$, is defined to be*

$$\inf\left\{\delta \in \mathbb{R}_{\geq 0} : \forall x \in \mathbb{R}_{\geq 0} \Pr_{a \leftarrow \mathcal{D}_0}[a \leq x] \leq \Pr_{a \leftarrow \mathcal{D}_1}[a \leq x + \varepsilon] + \delta.\right\}$$

We define the shift distance between two measurements $\mathcal{M}_0, \mathcal{M}_1$ over the same space $\mathcal{H}$ to be

$$\Delta^{\varepsilon}_{\mathsf{Shift}}(\mathcal{M}_0, \mathcal{M}_1) = \sup_{|\psi\rangle \in \mathcal{H}} \Delta_{\mathsf{Shift}}(\mathcal{M}_0|\psi\rangle, \mathcal{M}_1|\psi\rangle).$$

Definition 11 ((ε, δ)-Almost Projective [Zha20]). *Let Λ be a measurement with index set $\mathcal{I} \subseteq \mathbb{R}$. Λ is called* (ε, δ)-almost projective *if the following is satisfied for all states ρ of appropriate dimension. Apply Λ to ρ to obtain an outcome x and then apply Λ again to the post-measurement state to obtain an outcome x'. Then,* $\Pr[|x - x'| \leq \varepsilon] \geq 1 - \delta$.

Theorem 13 (Projective Implementation [Zha20]). *Let $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_0 = I - \mathcal{E}_1\}$ be a binary POVM. Then, there exists a projective measurement, called* projective implementation *of $\mathcal{E}$ and denoted $\mathsf{PI}(\mathcal{E})$, indexed by a finite set consisting of elements in $[0, 1]$ and it satisfies the following. For any state ρ of appropriate dimension, the following experiment has the same distribution as the outcome of applying $\mathcal{E}$ to ρ.*

1. *Apply $\mathsf{PI}(\mathcal{E})$ to ρ to obtain an outcome p.*
2. *Output 1 with probability p and 0 otherwise.*

Since $\mathsf{PI}(\mathcal{E})$ is projective, if the outcome of applying it to a state is p, then applying it again to the post-measurement state gives outcome p with probability 1.

Below, we will consider measurements that are defined as mixtures of projective measurements. For a collection of binary projective measurements $\mathcal{P} = \{P_i, I - P_i\}_{i \in \mathcal{I}}$ and a distribution $\mathcal{D}$ over $\mathcal{I}$, we will write $\mathcal{P}_{\mathcal{D}}$ to denote the measurement where we sample $i \leftarrow \mathcal{D}$ and apply the projective measurement $\{P_i, I - P_i\}$. In general, projective implementation of a mixture of projective measurements can be of exponential size, but it can be efficiently approximated.

[27] The original result is only for polynomial hardness against QPT adversaries, but it is easy to see that it also holds in the subexponential setting.

Theorem 14 (Approximate Projective Implementation [Zha20]). *Let $\mathcal{P} = \{P_i, I - P_i\}_{i\in\mathcal{I}}$ be a collection of binary projective measurements with index set $\mathcal{I}$ and $\mathcal{D}$ be a distribution over $\mathcal{I}$. Suppose we can efficiently implement the measurement*

$$\Lambda = \left\{\sum_{i\in\mathcal{I}}|i\rangle\langle i| \otimes P_i, I - \sum_{i\in\mathcal{I}}|i\rangle\langle i| \otimes P_i\right\}.$$

Then, for $0 < \varepsilon, \delta \le 1$, there exists a measurement, called approximate projective implementation *of $\mathcal{P}_{\mathcal{D}}$ and denoted $\mathsf{API}^{\varepsilon,\delta}_{\mathcal{P},\mathcal{D}}$, that satisfies the following.*

- $\mathsf{API}^{\varepsilon,\delta}_{\mathcal{P},\mathcal{D}}$ *is (ε, δ)-almost projective.*
- $\Delta^{\varepsilon}_{\mathsf{Shift}}(\mathsf{API}^{\varepsilon,\delta}_{\mathcal{P},\mathcal{D}}, \mathsf{PI}(\mathcal{P}_{\mathcal{D}})) \le \delta$.
- *Expected run time of $\mathsf{API}^{\varepsilon,\delta}_{\mathcal{P},\mathcal{D}}$ is polynomial in $1/\varepsilon, \log(1/\delta)$ and the runtimes of $\{P_i, I - P_i\}$, $\mathcal{D}$ and the procedure mapping i to $\{P_i, I - P_i\}$.*

Theorem 15 ([Zha20, Theorem 6.5]). *Let $\mathcal{D}_b$ for $b \in \{0, 1\}$ be efficient distributions over the same support with classical output and ρ be an efficiently constructible state. Let $\mathcal{P}$ be a collection of projective measurements indexed by the support of $\mathcal{D}_b$, and consider the mixture of measurements $\mathcal{P}_{\mathcal{D}_b}$ where we sample a measurement according to $\mathcal{D}_b$ and apply it. Suppose $\mathcal{D}_0 \approx \mathcal{D}_1$. Then, for any inverse polynomial ε,*

$$\Delta^{\varepsilon}_{\mathsf{Shift}}(\mathsf{PI}(\mathcal{P}_{\mathcal{D}_0}) \cdot \rho, \mathsf{PI}(\mathcal{P}_{\mathcal{D}_1}) \cdot \rho) \le \mathsf{negl}(\lambda).$$

We give the following generalization which differs from the previous theorem in a couple of aspects. First, we consider measurements over multiple registers. Second, we allow the measured state and the measurement to be correlated, which will be needed in our copy-protection proofs. Finally, we give more fine-grained results in terms of adversary's advantage and runtime, which will again be needed in our proofs.

Theorem 16. *Let λ denote the security parameter and let $k(\lambda)$ be a polynomial, $\varepsilon(\lambda)$ an inverse polynomial and $\delta(\lambda)$ be an inverse exponential.*

Let $\mathcal{S}^b$ and $\{\mathcal{B}^b_\ell\}_{\ell\in[k(\lambda)]}$ for each $b \in \{0, 1\}$ be efficient distributions as follows. $\mathcal{S}^b$ outputs a k-partite state and a classical string pp, while $\mathcal{B}^b_\ell$ take in pp and are classical. For each $\ell \in [k(\lambda)]$, consider the output distribution of the following experiment, denoted by $(\mathcal{S}^b, \mathcal{B}^b_\ell)$.

1. *$\rho, pp \leftarrow \mathcal{S}^b(1^\lambda)$.*
2. *Sample $s \leftarrow \mathcal{B}^b_\ell(pp)$.*
3. *Output (ρ, s, pp).*

Let $\mathcal{P}_\ell$ for each $\ell \in [k]$ be a collection of binary projective measurements indexed by output space of $\mathcal{B}^b_\ell$. For each fixed value of pp, consider the mixture of measurements, denoted $\mathcal{P}_{\ell,\mathcal{B}^b_\ell(pp)}$, where we sample a measurement s from $\mathcal{P}_\ell$ as $s = \mathcal{B}^b_\ell(pp; r)$ where $r \leftarrow \mathcal{R}$ and apply it. Suppose we can efficiently apply the above measurement for arbitrary given superpositions of r values. Let $\mathsf{API}^{\varepsilon,\delta}(\mathcal{P}_{\ell,\mathcal{B}^b_\ell(pp)})$ denote the approximate projective implementation of this mixture and let $\vec{p}_b$ be a tuple consisting of the outcomes of the following experiment.

1. *$\rho, pp \leftarrow \mathcal{S}^b(1^\lambda)$.*

2. *Apply* $\otimes_{\ell \in [k(\lambda)]} \mathsf{API}^{\varepsilon,\delta}(\mathcal{P}_{\ell,\mathcal{B}^b_\ell(pp)})$ *on* ρ.

Then,

- *Suppose* $(\mathcal{S}^0, \mathcal{B}^0_\ell) \approx (\mathcal{S}^1, \mathcal{B}^1_\ell)$ *for each* $\ell \in [k]$. *Then,*
$$|\vec{p}_0 - \vec{p}_1| \leq \mathsf{negl}(\lambda).$$
- *Suppose* $(\mathcal{S}^0, \mathcal{B}^0_\ell) \approx^c_{\nu(\lambda)} (\mathcal{S}^1, \mathcal{B}^1_\ell)$ *for all* $(\frac{k(\lambda)}{\mu^2(\lambda)} \cdot \mathsf{poly}(\lambda))$*-time adversaries for each* $\ell \in [k]$ *for some* ν, μ *satisfying* $\nu(\lambda) < \mu^2(\lambda)\mathsf{poly}(\lambda)$. *Then,*
$$|\vec{p}_0 - \vec{p}_1| \leq \mu(\lambda).$$

Proof. See Appendix A.5. □

Now, we reproduce the results of [ALL+21] regarding *threshold implementations.*

Theorem 17 (Threshold Implementation [ALL+21])**.** *Consider the following measurement, denoted* $\mathsf{ATI}^{\varepsilon,\delta}_{\mathcal{P},\mathcal{D},\eta}$, *associated with a collection of projective measurements* $\mathcal{P}$, *a distribution* $\mathcal{D}$ *over the index set of* $\mathcal{P}$ *and a threshold value* $\eta \in [0,1]$, *applied to a state* ρ.

1. *Apply* $\mathsf{API}^{\varepsilon,\delta}_{\mathcal{P},\mathcal{D}}$ *to* ρ, *let* p *be the outcome.*
2. *Outcome* 1 *if and only if* $p \geq \eta$.

We denote by $\mathrm{Tr}\left[\mathsf{ATI}^{\varepsilon,\delta}_{\mathcal{P},\mathcal{D},\eta} \cdot \rho\right]$ *the probability that the outcome above is* 1. *If* $\mathsf{API}^{\varepsilon,\delta}_{\mathcal{P},\mathcal{D}}$ *is replaced with* $\mathsf{PI}(\mathcal{P}_\mathcal{D})$, *then we denote the resulting measurement as* $\mathsf{TI}_\eta(\mathcal{P}_\mathcal{D})$ *and write* $\mathrm{Tr}[\mathsf{TI}_\eta(\mathcal{P}_\mathcal{D}) \cdot \rho]$ *to denote the probability that the outcome is* 1.

We then have the following.

- *For any state* ρ,
$$\mathrm{Tr}\left[\mathsf{ATI}^{\varepsilon,\delta}_{\mathcal{P},\mathcal{D},\eta-\varepsilon} \cdot \rho\right] \geq \mathrm{Tr}[\mathsf{TI}_\eta(\mathcal{P}_\mathcal{D}) \cdot \rho] - \delta.$$
- *For any state* ρ,
$$\mathrm{Tr}[\mathsf{TI}_{\eta-\varepsilon}(\mathcal{P}_\mathcal{D}) \cdot \rho] \geq \mathrm{Tr}\left[\mathsf{ATI}^{\varepsilon,\delta}_{\mathcal{P},\mathcal{D},\eta} \cdot \rho\right] - \delta.$$
- $\mathsf{ATI}^{\varepsilon,\delta}_{\mathcal{P},\mathcal{D},\eta}$ *is efficient whenever* $\mathsf{API}^{\varepsilon,\delta}_{\mathcal{P},\mathcal{D}}$ *is.*
- $\mathsf{TI}_{\mathcal{P},\mathcal{D},\eta}$ *is a projection and the collapsed state conditioned on outcome* 1 *is a mixture of eigenvectors of* $\mathcal{D}$ *with eigenvalue* $\geq \eta$.

We give some further generalizations below.

Theorem 18. *For any* $k \in \mathbb{N}$, *let* $\mathcal{P}_\ell, \mathcal{D}_\ell$ *be a collection of projective measurements and a distribution on the index set of this collection, respectively, and* $\eta_\ell \in [0,1]$ *be threshold values for all* $\ell \in [k]$. *Write* $\mathrm{Tr}\left[\left(\bigotimes_{\ell \in [k]} \mathsf{ATI}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{D}_\ell,\eta_\ell}\right) \cdot \rho\right]$ *to denote the probability that the outcome of the joint measurement* $\bigotimes_{\ell \in [k]} \mathsf{ATI}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{D}_\ell,\eta_\ell}$ *applied on* ρ *is all* 1, *and similarly for* TI.

Then, we have the following.

- *[ALL+20, Corollary 3] For any k-partite state ρ,*

$$\mathrm{Tr}\left[\left(\bigotimes_{\ell\in[k]}\mathsf{ATI}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{D}_\ell,\eta_\ell-\varepsilon}\right)\rho\right]\geq\mathrm{Tr}\left[\left(\bigotimes_{\ell\in[k]}\mathsf{TI}_{\eta_\ell}(\mathcal{P}_{\ell\mathcal{D}_\ell})\right)\rho\right]-k\cdot\delta.$$

- *[ALL+20, Corollary 3] For any k-partite state ρ, let ρ' be the collapsed state obtained after applying $\bigotimes_{\ell\in[k]}\mathsf{ATI}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{D}_\ell,\eta_\ell}$ to ρ and obtaining the outcome 1. Then,*

$$\mathrm{Tr}\left[\left(\bigotimes_{\ell\in[k]}\mathsf{TI}_{\eta_\ell-2\varepsilon}(\mathcal{P}_{\ell\mathcal{D}_\ell})\right)\rho'\right]\geq 1-2k\cdot\delta.$$

- *For any k-partite state ρ, let ρ' be the collapsed state obtained after applying $\bigotimes_{\ell\in[k]}\mathsf{ATI}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{D}_\ell,\eta}$ to ρ and obtaining the outcome 1. Then,*

$$\mathrm{Tr}\left[\left(\bigotimes_{\ell\in[k]}\mathsf{ATI}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{D}_\ell,\eta_\ell-3\varepsilon}\right)\cdot\rho'\right]\geq 1-3k\cdot\delta.$$

- *For any k-partite state ρ,*

$$\mathrm{Tr}\left[\left(\bigotimes_{\ell\in[k]}\mathsf{TI}_{\eta_\ell-\varepsilon}(\mathcal{P}_{\ell\mathcal{D}_\ell})\right)\rho\right]\geq\mathrm{Tr}\left[\left(\bigotimes_{\ell\in[k]}\mathsf{ATI}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{D}_\ell,\eta_\ell}\right)\rho\right]-k\cdot\delta.$$

Proof. See Appendix A.6. □

Theorem 19. *For any $k\in\mathbb{N}$, let $\mathcal{P}_\ell,\mathcal{D}_\ell$ be a collection of projective measurements and a distribution on the index set of this collection, respectively. Let ρ be any k-partite state of appropriate dimension. Consider the measurement outcome $\vec{p}$ and the post-measurement state ρ' obtained by applying $\left(\bigotimes_{\ell\in[k]}\mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{D}_\ell}\right)$ to a state ρ. Let $\vec{p'}$ be the measurement outcome obtained by applying the measurement $\left(\bigotimes_{\ell\in[k]}\mathsf{PI}(\mathcal{P}_{\ell\mathcal{D}_\ell})\right)$ to ρ'. Then,*

$$\Pr\Big[\forall\ell\in[k]\quad(\vec{p'})_\ell\leq(\vec{p})_\ell+2\varepsilon\Big]\geq 1-2\cdot k\cdot\delta$$
$$\Pr\Big[\forall\ell\in[k]\quad(\vec{p'})_\ell\geq(\vec{p})_\ell-2\varepsilon\Big]\geq 1-2\cdot k\cdot\delta.$$

Proof. See Appendix A.6. □

Theorem 20. *For any $k\in\mathbb{N}$, let $\mathcal{P}_\ell,\mathcal{D}_\ell$ be a collection of projective measurements and a distribution on the index set of this collection, respectively. Let ρ be any k-partite state of appropriate dimension. Consider the measurement outcome $\vec{p}$ obtained by applying $\left(\bigotimes_{\ell\in[k]}\mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{D}_\ell}\right)$ to a state ρ. Let $\vec{p'}$ be the measurement outcome obtained by applying the measurement $\left(\bigotimes_{\ell\in[k]}\mathsf{PI}(\mathcal{P}_{\ell\mathcal{D}_\ell})\right)$ to ρ. Then,*

$$\Pr\Big[\forall\ell\in[k]\quad(\vec{p'})_\ell\leq\eta_\ell+\varepsilon\Big]\geq\Pr[\forall\ell\in[k]\quad(\vec{p})_\ell\leq\eta_\ell]-k\cdot\delta$$
$$\Pr[\forall\ell\in[k]\quad(\vec{p})_\ell\leq\eta_\ell+\varepsilon]\geq\Pr\Big[\forall\ell\in[k]\quad(\vec{p'})_\ell\leq\eta_\ell\Big]-k\cdot\delta.$$

Proof. See Appendix A.6. □

5 Coset States

In this section, we start by giving the definition of coset states [CLLZ21], which are the states we use in our constructions, and then state the monogamy-of-entanglement property they satisfy. Then, we define two new security games for coset states that streamlines our proofs later on and show secure constructions for these games.

Definition 12 (Coset States [CLLZ21]). *Let A be a subspace of $\mathbb{F}_2^n$ and s, s' be vectors in $\mathbb{F}_2^n$. We define the coset state associated with A, s, s', denoted $|A_{s,s'}\rangle$, to be*

$$|A_{s,s'}\rangle = \sum_{a \in A} \frac{1}{\sqrt{|A|}} (-1)^{\langle s', a\rangle} |a + s\rangle.$$

We will write $A + s$ to denote both the coset $A + s$ and the program that takes as input a vector $v \in \mathbb{F}_2^n$ and outputs 1 if and only if $v \in A + s$, and 0 otherwise. The distinction will be clear from context.

Theorem 21 ([CLLZ21]). *Consider a subspace $A \subseteq \mathbb{F}_2^n$ and vectors $s, s' \in \mathbb{F}_2^n$.*

1. *There exists an efficient quantum algorithm that outputs $|A_{s,s'}\rangle$ given s, s' and the description of A.*
2. *$H^{\otimes n}|A_{s,s'}\rangle = |(A^\perp)_{s',s}\rangle$.*
3. *We define the canonical element of a coset $A + v$ to be the lexicographically smallest element in the coset and denote it $\mathsf{Can}_A(v)$. There exists an efficient classical algorithm that, on input the description of A and a vector v, outputs $\mathsf{Can}_A(v)$.*

Coset states satisfy a natural monogamy-of-entanglement (*MoE*) property where any adversary can win the following game with only negligible probability: We present the adversary with a coset state, and it is required to split the state into two (possibly entangled) registers that can be used to simultaneously output vectors in the cosets $A + s$ and $A^\perp + s'$ respectively.

Theorem 22 (Monogamy-of-Entanglement Property for Coset States [CLLZ21, CV22]). *Consider the following game between the challenger and an adversary tuple $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$.*

$\underline{\mathsf{MoE}(\lambda, \mathcal{A})}$

1. *Sample uniformly at random a subspace A of $\mathbb{F}_2^\lambda$ of dimension $\frac{\lambda}{2}$ and two elements $s, s' \leftarrow \mathbb{F}_2^\lambda$.*
2. *Sample $\mathsf{OP}^0 \leftarrow i\mathcal{O}(A + s)$ and $\mathsf{OP}^1 \leftarrow i\mathcal{O}(A^\perp + s')$.*
3. *Submit $|A_{s,s'}\rangle, \mathsf{OP}^0, \mathsf{OP}^1$ to $\mathcal{A}_0$.*
4. *$\mathcal{A}$ outputs two (possibly entangled) registers $\mathsf{R}_1, \mathsf{R}_2$.*
5. *For $j \in \{1, 2\}$, run $v_j \leftarrow \mathcal{A}_j(\mathsf{R}_j, A)$.*
6. *Output 1 if and only if $v_1 \in A + s$ and $v_2 \in A^\perp + s'$.*

Assuming the existence of $i\mathcal{O}$ and one-way functions, then for any QPT adversary tuple $\mathcal{A}$,

$$\Pr[\mathsf{MoE}(\lambda, \mathcal{A}) = 1] \leq \mathsf{negl}(\lambda).$$

If we assume the existence of subexponentially-secure $i\mathcal{O}$ and one-way functions, then there exists a constant $C_{\mathsf{MoE}} > 0$ such that for any QPT adversary tuple

$$\Pr[\mathsf{MoE}(\lambda, \mathcal{A}) = 1] \leq 2^{-\lambda^{C_{\mathsf{MoE}}}}$$

for all sufficiently large λ.

In the previous constructions of unclonable primitives [CLLZ21, LLQZ22], and also in our constructions, the security of the unclonable schemes rely on requiring the *freeloader* adversaries to output a vector from either $A + s$ or $A^\perp + s'$, depending on a random challenge bit presented to them. However, the *pirate* (i.e. splitting) adversary can always guess this challenge bit and measure the coset state accordingly before splitting into freeloaders, and it would be right for both freeloaders with probability $(1/2)^2$. Therefore, to achieve negligible or subexponential security, we amplify the security by using multiple coset states. This variant of the game is implicitly used in [CLLZ21, LLQZ22] and a similar amplification theorem for a related game is also formally proven in [ÇGLZR24]; both for the case of uniformly random challenge strings. We generalize the amplification result to any unpredictable distribution of challenge strings.

Definition 13 (Unpredictable Distribution)**.** *Let $\mathcal{D} = \{\mathcal{D}_\lambda\}_\lambda$ be a family of distributions over $\{0,1\}^{a(\lambda)}$ where $a(\cdot)$ is some polynomial and we write $\mathcal{D}(x)$ to mean $\Pr_{x' \leftarrow \mathcal{D}}[x' = x]$. Then, $\mathcal{D}$ is said to be (statistically)* unpredictable *if $\max_{x \in \{0,1\}^{a(\lambda)}} \{\mathcal{D}(x)\} \leq \mathsf{negl}(\lambda)$.*

Definition 14. *Define $\mathsf{CosetGen}$ to be the following algorithm, where we have the default parameter values $a(\lambda) = 3 \cdot \lambda^3$ and $\kappa(\lambda) = \lambda^{\lceil 3/C_{\mathsf{MoE}} \rceil}$.*

```
$\mathsf{CosetGen}(1^\lambda, a(\lambda) = 3 \cdot \lambda^3, \kappa(\lambda) = \lambda^{\lceil 3/C_{\mathsf{MoE}} \rceil})$

  1. For $i \in [a(\lambda)]$, sample uniformly at random a subspace $A_i$ of $\mathbb{F}_2^{\kappa(\lambda)}$ of dimension $\kappa(\lambda)/2$
     and two elements $s_i, s'_i \leftarrow \mathbb{F}_2^{\kappa(\lambda)}$.
  2. Output $(A_i, s_i, s'_i)_{i \in [a(\lambda)]}$.
```

We call the output of $\mathsf{CosetGen}$ a coset tuple.

Theorem 23 (Monogamy-of-Entanglement Property for Coset States - Multi-Challenge Version)**.** *Let $\mathcal{D}$ be a distribution over $\{0,1\}^{a(\lambda)}$ that is unpredictable. Consider the following game between the challenger and an adversary tuple $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$.*

```
$\mathsf{MoE} - \mathsf{MultChal}(\lambda, \mathcal{A})$

  1. $(A_i, s_i, s'_i)_{i \in [a(\lambda)]} \leftarrow \mathsf{CosetGen}(1^\lambda, a(\lambda), \kappa(\lambda))$.
  2. For $i \in [a(\lambda)]$,
     2.1. Sample $\mathsf{OP}_i^0 \leftarrow i\mathcal{O}(A_i + s_i)$.
     2.2. Sample $\mathsf{OP}_i^1 \leftarrow i\mathcal{O}(A_i^\perp + s'_i)$.
```

3. *Submit* $\left\{\left|A_{i,s_i,s'_i}\right\rangle\right\}_{i\in[a(\lambda)]}, (\mathsf{OP}^0_i, \mathsf{OP}^1_i)_{i\in[a(\lambda)]}$ *to* $\mathcal{A}_0$*.*

4. $\mathcal{A}$ *outputs two (possibly entangled) registers* $\mathsf{R}_1, \mathsf{R}_2$*.*

5. *Challenger samples* $r_1 \leftarrow \mathcal{D}$ *and* $r_2 \leftarrow \mathcal{D}$*.*

6. *For* $\ell \in \{1,2\}$*, run* $(v_{\ell,i})_{i\in[a(\lambda)]} \leftarrow \mathcal{A}_j(\mathsf{R}_j, r_j, (A_i)_{i\in[a(\lambda)]})$*.*

7. *For* $\ell \in \{1,2\}$ *and all* $i \in [a(\lambda)]$*, check if* $v_{\ell,i} \in A_i + s_i$ *if* $(r_\ell)_i = 0$ *and if* $v_{\ell,i} \in A_i^\perp + s'_i$ *if* $(r_\ell)_i = 1$*. Output* 1 *if and only if all the checks pass. Otherwise, output* 0*.*

Assuming the existence of $i\mathcal{O}$ *and one-way functions, and setting* $\kappa(\lambda) = \lambda$*, then for any QPT adversary tuple* $\mathcal{A}$*,*

$$\Pr[\mathsf{MoE} - \mathsf{MultChal}(\lambda, \mathcal{A}) = 1] \leq \mathsf{negl}(\lambda).$$

If we assume the existence of subexponentially-secure $i\mathcal{O}$ *and one-way functions, and set* $\mathcal{D}$ *to be a distribution that is subexponentially unpredictable, then there exists a constant* $C_{\mathsf{MoE.MultChal}}$ *such that for any QPT adversary tuple*

$$\Pr[\mathsf{MoE} - \mathsf{MultChal}(\lambda, \mathcal{A}) = 1] \leq 2^{-\lambda^{C_{\mathsf{MoE.MultChal}}}}$$

for all sufficiently large λ*. By setting* $\mathcal{D}$ *to be* $2^{-3\cdot\lambda^3}$ *unpredictable and* $\kappa(\lambda) = \lambda^{\lceil 3/C_{\mathsf{MoE}}\rceil}$*, there exists such* $C_{\mathsf{MoE.MultChal}} > 2$*.*

Proof. Suppose there exists an QPT adversary tuple $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ that wins $\mathsf{MoE} - \mathsf{MultChal}$ above with probability $\varepsilon(\lambda)$, that is, $\Pr[\mathsf{MoE} - \mathsf{MultChal}(\lambda, \mathcal{A}) = 1] \geq \varepsilon(\lambda)$.

Consider the following adversary $\mathcal{A}' = (\mathcal{A}'_0, \mathcal{A}'_1, \mathcal{A}'_2)$ for MoE.

$\underline{\mathcal{A}'_0}$

On input a state ρ and the obfuscated programs $\mathsf{OP}^{0*}, \mathsf{OP}^{1*}$, sample $r_1 \leftarrow \mathcal{D}$ and $r_2 \leftarrow \mathcal{D}$. If $r_1 = r_2$, then abort. Let j^* be an index where r_1 and r_2 differ, that is, $(r_1)_{j^*} \neq (r_2)_{j^*}$. For all $j \in [a(\lambda)] \setminus \{j^*\}$, sample a subspace A_j, elements $s_j, s'_j \leftarrow \mathbb{F}_2^{\kappa(\lambda)}$, then set $\rho_j = \left|A_{j,s_j,s'_j}\right\rangle$ and sample $\mathsf{OP}^0_j \leftarrow i\mathcal{O}(A_j + s_j)$ and $\mathsf{OP}^1_j \leftarrow i\mathcal{O}(A_j^\perp + s'_j,)$. Then, run $\mathcal{A}((\rho_j, \mathsf{OP}^0_j, \mathsf{OP}^1_j)_{j\in[a(\lambda)]})$ to obtain a bipartite state σ. Finally, output

$$((\sigma[1], (A_j)_{j\in[a(\lambda)]\setminus\{j^*\}}, j^*, r_1), (\sigma[2], (A_j)_{j\in[a(\lambda)]\setminus\{j^*\}}, j^*, r_2)).$$

if $(r_1)_{j^*} = 0$ and $(r_2)_{j^*} = 1$, or

$$((\sigma[2], (A_j)_{j\in[a(\lambda)]\setminus\{j^*\}}, j^*, r_2), (\sigma[1], (A_j)_{j\in[a(\lambda)]\setminus\{j^*\}}, j^*, r_1)).$$

if $(r_1)_{j^*} = 1$ and $(r_2)_{j^*} = 0$.

$\underline{\mathcal{A}'_\ell \textbf{ for } \ell \in \{1,2\}}$

$\mathcal{A}'_\ell$ runs $\mathcal{A}_\ell$ on its own input and the subspace description A it obtains from the challenger of $\mathsf{MoE} - \mathsf{MultChal}$. Note that $\mathcal{A}'_\ell$ can correctly rearrange the input order when passing it to $\mathcal{A}_\ell$ since it knows j^*. Finally, it outputs the j^*-th vector in the output of $\mathcal{A}_\ell$.

It is easy to see that the adversary $\mathcal{A}'$ wins whenever it does not abort and the vectors output by $\mathcal{A}$ are correct. Let p_{s_1,s_2} denote the probability of $\mathcal{A}$ winning $\mathsf{MoE} - \mathsf{MultChal}$ conditioned on $r_1 = s_1$ and $r_2 = s_2$. Then, we have

$$\varepsilon(\lambda) = \sum_{s_1,s_2\in\{0,1\}^{a(\lambda)}} \mathcal{D}(s_1) \cdot \mathcal{D}(s_2) p_{s_1,s_2}$$

and

$$\begin{aligned}\Pr\big[\mathsf{MoE}(\lambda,\mathcal{A}')\big] &= \sum_{s_1\neq s_2\in\{0,1\}^{a(\lambda)}} \mathcal{D}(s_1)\cdot\mathcal{D}(s_2)\cdot p_{s_1,s_2}\\ &= \varepsilon(\lambda) - \sum_{s\in\{0,1\}^{a(\lambda)}} (\mathcal{D}(s))^2\cdot p_{s,s}\\ &\geq \varepsilon(\lambda) - \sum_{s\in\{0,1\}^{a(\lambda)}} (\mathcal{D}(s))^2\\ &\geq \varepsilon(\lambda) - \left[\sum_{s\in\{0,1\}^{a(\lambda)}} (\mathcal{D}(s))\right]\cdot \max_{s\in\{0,1\}^{a(\lambda)}}\{\mathcal{D}(s)\}\\ &\geq \varepsilon(\lambda) - \max_{s\in\{0,1\}^{a(\lambda)}}\{\mathcal{D}(s)\}.\end{aligned}$$

For the case of negligible security, we will have $\varepsilon(\lambda) > \frac{1}{\mathsf{poly}(\lambda)}$ and $\max_{s\in\{0,1\}^{a(\lambda)}}\{\mathcal{D}(s)\} = \mathsf{negl}(\lambda)$ since $\mathcal{D}$ is unpredictable, hence we get $\Pr[\mathsf{MoE}(\lambda,\mathcal{A}')] > \frac{1}{\mathsf{poly}(\lambda)}$, which is a contradiction by Theorem 22. The subexponential cases follow by similar calculations. □

Finally, we introduce another variant of the game that is useful for our unbounded collusion secure constructions. In this game, the adversary queries multiple coset state tuples (which are associated with *identity strings*) that are generated pseudorandomly, and it is allowed to choose the coset state tuple for which it wants to *break* the monogamy-of-entanglement property. The adversary is also presented with an (obfuscated) program that allows it to make membership queries for any coset tuple by specifying its identity.

Theorem 24 (Monogamy-of-Entanglement Property for Coset States - Collusion-Resistant Version)**.** *Let $L(\lambda)$ be a polynomial, denoting the length of the identity strings. Define $c_L(\lambda) = 3\cdot(L(\lambda)+\lambda)^3$. Let $\mathcal{D}$ be a distribution over $\{0,1\}^{c_L(\lambda)}$. Consider the following game between the challenger and an adversary tuple $\mathcal{A} = (\mathcal{A}_0,\mathcal{A}_1,\mathcal{A}_2)$.*

$\underline{\mathsf{MoE}-\mathsf{Coll}(\lambda, L(\lambda), \mathcal{A})}$

1. *The challenger initializes the list* $\mathsf{ID} = [\,]$.
2. *The challenger samples a PRF key* $K \leftarrow F.\mathsf{KeyGen}(1^\lambda)$.
3. *The challenger samples* $\mathsf{OPMem} \leftarrow i\mathcal{O}(\mathsf{PMem}_K)$*, where* PMem_K *is the following program.*

```
$\underline{\mathsf{PMem}_K(id, u_1, \ldots, u_{c_L(\lambda)}, r)}$
Hardcoded: $K$
  1. $(A_i, s_i, s'_i)_{i\in[c_L(\lambda)]} \leftarrow \mathsf{CosetGen}(1^{L(\lambda)+\lambda}; F(K, id))$.
  2. For each $i \in [c_L(\lambda)]$, check if $u_i \in A_i + s_i$ if $(r)_i = 0$ and check if $u_i \in A_i^\perp + s'_i$ if
     $(r)_i = 1$. If any of the checks fail, output $0$ and terminate.
  3. Output $1$.
```

4. *The challenger submits* OPMem *to the adversary.*

5. ***Query Phase 1:*** *For polynomially many rounds, the adversary makes queries as follows. The adversary submits an identity string id* $\in \{0,1\}^{L(\lambda)}$ *to the challenger. Then, the challenger adds id to the list* ID, *samples* $(A_i, s_i, s'_i)_{i\in[c_L(\lambda)]} \leftarrow \mathsf{CosetGen}(1^{L(\lambda)+\lambda}; F(K, id))$ *and submits the state* $\left\{\left|A_{i,s_i,s'_i}\right\rangle\right\}_{i\in[c_L(\lambda)]}$ *to the adversary.*

6. ***Splitting Phase:*** *The adversary* $\mathcal{A}_0$ *outputs outputs an identity string id*$^* \in \{0,1\}^{L(\lambda)}$ *and a* bipartite *register* R.

7. *The challenger samples* $(A_i^*, s_i^*, s_i^{'*})_{i\in[c_L(\lambda)]} \leftarrow \mathsf{CosetGen}(1^{L(\lambda)+\lambda}; F(K, id^*))$.

8. ***Query Phase 2:*** *For* $\ell \in \{1,2\}$*, each adversary* $\mathcal{A}_\ell$ *is given* $R[\ell]$ *and* $(A_i^*)_{i\in[c_L(\lambda)]}$. *For polynomially many rounds, each adversary makes queries to the challenger as follows.* $\mathcal{A}_\ell$ *submits an identity string id to the challenger. If id* $\neq$ *id***, the challenger samples* $(A_i, s_i, s'_i)_{i\in[c_L(\lambda)]} \leftarrow$ $\mathsf{CosetGen}(1^{L(\lambda)+\lambda}; F(K, id))$ *and submits the state* $\left\{\left|A_{i,s_i,s'_i}\right\rangle\right\}_{i\in[c_L(\lambda)]}$ *to the adversary* $\mathcal{A}_\ell$.

9. ***Challenge Phase:*** *The challenger samples* $r_1 \leftarrow \mathcal{D}$ *and* $r_2 \leftarrow \mathcal{D}$.

10. *For* $\ell \in \{1,2\}$*, each adversary* $\mathcal{A}_\ell$ *is given* r_ℓ *and it outputs a tuple of vectors* $(v_{\ell,i})_{i\in[c_L(\lambda)]}$.

11. *The challenger, for all* $\ell \in \{1,2\}$ *and for all* $i \in [c_L(\lambda)]$*, checks if* $v_{\ell,i} \in A_i^* + s_i^*$ *if* $(r_\ell)_i = 0$ *and checks if* $v_{\ell,i} \in (A^*)_i^\perp + s_i^{'*}$ *if* $(r_\ell)_i = 1$.

    If all the checks above pass and id* *appears in* ID *at most once, the challenger outputs* 1. *Otherwise, it outputs* 0.

Similarly, we define $\mathsf{MoE} - \mathsf{Coll} - \mathsf{Sel}(\lambda, L(\lambda), \mathcal{A})$ *to be the selective version of the above game where the adversary outputs the chosen identity id** *at the beginning of the game.*

Assuming the existence of $i\mathcal{O}$ *and one-way functions, then for any polynomial* $L(\lambda)$*, for any unpredictable distribution* $\mathcal{D}$ *and for any QPT adversary tuple* $\mathcal{A}$*,*

$$\Pr[\mathsf{MoE} - \mathsf{Coll} - \mathsf{Sel}(\lambda, L(\lambda), \mathcal{A}) = 1] \leq \mathsf{negl}(\lambda).$$

If we assume the existence of subexponentially-secure $i\mathcal{O}$ *and one-way functions, and set* $\mathcal{D}$ *to be the uniform distribution, then for any polynomial* $L(\lambda)$ *there exists constants* $C_{\mathsf{MoE.Coll.Sel}}, C_{\mathsf{MoE.Coll}} > 0$ *such that for any QPT adversary tuple*

$$\Pr[\mathsf{MoE} - \mathsf{Coll} - \mathsf{Sel}(\lambda, L(\lambda), \mathcal{A}) = 1] \leq 2^{-\lambda^{C_{\mathsf{MoE.Coll.Sel}}}}$$

$$\Pr[\mathsf{MoE} - \mathsf{Coll}(\lambda, L(\lambda), \mathcal{A}) = 1] \leq 2^{-\lambda^{C_{\mathsf{MoE.Coll}}}}$$

for all sufficiently large λ.

Proof. We will give the full security proof for the *adaptive* case. For the adaptive case, we will rely on *complexity leveraging*, i.e. basically guessing the challenge identity *id**. The security of the selective game $\mathsf{MoE} - \mathsf{Coll} - \mathsf{Sel}$ follows from the same arguments, with the difference being that we do not use complexity leveraging, since the *id** is selected by the adversary at the beginning (hence no need to guess it), and therefore we can simply puncture the PRF key at that point when preparing OPMem.

We now prove adaptive security. Let $i\mathcal{O}$ be a $2^{-\lambda^{c_{\mathsf{iO}}}}$-secure indistinguishability obfuscation scheme and F be a $2^{-\lambda^{c_{\mathsf{PRF}}}}$-secure puncturable PRF family with input length $L(\lambda)$ and output length same as the size of the randomness used by $\mathsf{CosetGen}$, where $c_{\mathsf{iO}}, c_{\mathsf{PRF}}$ are some constants satisfying $\lambda^{c_{\mathsf{PRF}}} > (\lambda + L(\lambda))^3$ and $\lambda^{c_{\mathsf{iO}}} > (\lambda + L(\lambda))^3$. Note that such a PRF exists assuming subexponentially secure one-way functions (Theorem 6).

We first define a stronger game as follows.

$\underline{\mathsf{Moe-Coll-PuncKey}}$

1. The challenger initializes the list $\mathsf{ID} = [\,]$.
2. The challenger samples a PRF key $K \leftarrow F.\mathsf{KeyGen}(1^\lambda)$.
3. The challenger samples $\mathsf{OPMem} \leftarrow i\mathcal{O}(\mathsf{PMem}_K)$, where PMem_K is the following program.

```
$\underline{\mathsf{PMem}_K(id, u_1, \ldots, u_{c_L(\lambda)}, r)}$
**Hardcoded:** $K$
  1. $(A_i, s_i, s_i')_{i\in[c_L(\lambda)]} \leftarrow \mathsf{CosetGen}(1^{L(\lambda)+\lambda}; F(K, id))$.
  2. For each $i \in [c_L(\lambda)]$, check if $u_i \in A_i + s_i$ if $(r)_i = 0$ and check if $u_i \in A_i^\perp + s_i'$ if $(r)_i = 1$. If any of the checks fail, output 0 and terminate.
  3. Output 1.
```

4. The challenger submits OPMem to the adversary.
5. For polynomially many rounds, the adversary makes queries as follows. The adversary submits an identity string id to the challenger and a query type, either $\mathsf{CLASSICAL}$ or STATE. Then, the challenger samples $(A_i, s_i, s_i')_{i\in[c_L(\lambda)]} \leftarrow \mathsf{CosetGen}(1^{L(\lambda)+\lambda}; F(K, id))$.

   If the type is $\mathsf{CLASSICAL}$, the challenger adds id to the list ID *twice* and submits $(A_i, s_i, s_i')_{i\in[c_L(\lambda)]}$ to the adversary.

   If the type is STATE, the challenger adds id to the list ID *once* and submits the state $\left\{\left|A_{i,s_i,s_i'}\right\rangle\right\}_{i\in[c_L(\lambda)]}$ to the adversary.
6. The adversary $\mathcal{A}_0$ outputs outputs an identity string $id^* \in \{0,1\}^{L(\lambda)}$.
7. The challenger computes $(A_i^*, s_i^*, s_i'^*)_{i\in c_L(\lambda)} = \mathsf{CosetGen}(1^{L(\lambda)+\lambda}; F(K, id^*))$.
8. The challenger computes $K\{id^*\} \leftarrow F.\mathsf{Punc}(K, id^*)$ and submits it to the adversary.
9. The adversary outputs a *bipartite* register R.
10. The challenger samples $r_1 \leftarrow \mathcal{D}$ and $r_2 \leftarrow \mathcal{D}$.
11. For $\ell \in \{1, 2\}$, each adversary $\mathcal{A}_\ell$ is given $R[\ell], (A_i^*)_{i\in[c_L(\lambda)]}, r_\ell$ and $K\{id^*\}$, and it outputs a tuple of vectors $(v_{\ell,i})_{i\in[c_L(\lambda)]}$.
12. The challenger, for all $\ell \in \{1, 2\}$ and for all $i \in [c_L(\lambda)]$, checks if $v_{\ell,i} \in A_i^* + s_i^*$ if $(r_\ell)_i = 0$ and checks if $v_{\ell,i} \in (A^*)_i^\perp + s_i'^*$ if $(r_\ell)_i = 1$.

    If all the checks above pass and id^* appears in ID at most once, the challenger outputs 1. Otherwise, it outputs 0.

It is easy to see that the security in the stronger game implies security in the original game, since the adversaries $\mathcal{A}_1, \mathcal{A}_2$ can simulate their coset queries simply by evaluating the PRF using $K\{id^*\}$. Note that in the original game, they are not allowed to query for id^* after the split, therefore $K\{id^*\}$ rather than K is sufficient.

Now suppose for a contradiction that there exists an adversary $(\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ that wins the stronger security game with probability $2^{-\lambda}$. We define a tuple of efficient algorithms $(\mathcal{A}_0', \mathcal{A}_1', \mathcal{A}_2')$ as follows.

$\underline{\mathcal{A}_0'}$

Sample $id' \leftarrow \{0,1\}^{L(\lambda)}$. Simulate both $\mathcal{A}_0$ and the challenger of the stronger game above, up to (including) Item 8. If $id^* = id'$, output the output of $\mathcal{A}_0$, along with two copies of $(K\{id^*\}, (A_i^*)_{i \in [c_L(\lambda)]})$, one for each $\mathcal{A}_\ell'$. Otherwise, output $(\bot, \bot)$.

$\underline{\mathcal{A}_\ell'}$ **for** $\underline{\ell \in \{1,2\}}$

If the input is $\bot$, output $\bot$ and terminate. Otherwise, simulate the rest of the challenger and $\mathcal{A}_\ell$.

Observe that the probability that both $\mathcal{A}_\ell'$ simultaneously output the *correct* vectors is at least $2^{-\lambda} \cdot 2^{-L(\lambda)}$, since $\mathcal{A}$ outputs the correct vectors with probability $2^{-\lambda}$ by assumption and we have $id' = id^*$ with probability $2^{-L(\lambda)}$ independently. Now, we will modify the algorithms $\mathcal{A}'$ through a sequence of steps to finally obtain an adversary that wins $\mathsf{MoE} - \mathsf{MultChal}$ with probability $2^{-2\cdot(\lambda+L(\lambda))}$, which is a contradiction by Theorem 23. Throughout rest of the proof, we will assume $id^* = id'$, which is indeed required to win the game.

We define $\mathcal{A}_0''$ by modifying $\mathcal{A}_0'$ so that it now samples OPMem as follows. It computes $z = F(K, id')$ at the beginning of the game and $(A_i^*, s_i^*, s_i'^*)_{i \in c_L(\lambda)} = \mathsf{CosetGen}(1^{L(\lambda)+\lambda}; F(K, id^*))$. Then, we first compute for each $i \in [c_L(\lambda)]$, $\mathsf{OP}_i^{*0} \leftarrow i\mathcal{O}(A_i^* + s_i^*)$ and $\mathsf{OP}_i^{*1} \leftarrow i\mathcal{O}(A_i^{*\perp} + s_i'^*)$ (i.e., as in Theorem 23) using $(A_i^*, s_i^*, s_i'^*)_{i \in c_L(\lambda)}$. Then, it samples $\mathsf{OPMem} \leftarrow i\mathcal{O}(\mathsf{PMem'}_{K\{id'\}, id', (\mathsf{OP}_i^{*0}, \mathsf{OP}_i^{*1})_{i \in [c_L(\lambda)]}})$.

```
$\underline{\mathsf{PMem'}_{K\{id'\}, id', (\mathsf{OP}_i^{*0}, \mathsf{OP}_i^{*1})_{i \in [c_L(\lambda)]}}(id, u_1, \dots, u_{c_L(\lambda)}, r)}$
    Hardcoded: $K\{id'\}, id', (\mathsf{OP}_i^{*0}, \mathsf{OP}_i^{*1})_{i \in [c_L(\lambda)]}$
    1. If $id = id'$, execute the following.
        1.1. For each $i \in [c_L(\lambda)]$, check if $\mathsf{OP}_i^{*0}(u_i) = 1$ if $(r)_i = 0$ and check if $\mathsf{OP}_i^{*1}(u_i) = 1$ if $(r)_i = 1$.
        1.2. If all the checks pass, output 1 and terminate. Otherwise, output 0 and terminate.
    2. $(A_i, s_i, s_i')_{i \in [c_L(\lambda)]} \leftarrow \mathsf{CosetGen}(1^{L(\lambda)+\lambda}; F_1(K\{id'\}, id))$.
    3. For each $i \in [c_L(\lambda)]$, check if $u_i \in A_i + s_i$ if $(r)_i = 0$ and check if $u_i \in A_i^\perp + s_i'$ if $(r)_i = 1$.
       If any of the checks fail, output 0 and terminate.
    4. Output 1.
```

Further, $\mathcal{A}_0''$ answers any query made by $\mathcal{A}$ for id' using z. By correctness of the obfuscations $\mathsf{OP}_i^{*0}, \mathsf{OP}_i^{*1}$, and by security of the obfuscation of $\mathsf{PMem'}$, we get that the modified adversary outputs the correct vectors with probability at least $2^{-\lambda-L(\lambda)} - 2^{-(\lambda+L(\lambda))^3}$.

Observe that above, we never evaluate the PRF at id' except at the first step, where we compute z, and the adversary only gets the punctured key $K\{id'\}$[28]. We define $\mathcal{A}_0''$ so that it now samples z uniformly at random. Then, by above and by puncturable PRF security (Definition 1), the adversary outputs the correct vectors with probability at least $2^{-\lambda-L(\lambda)} - 2 \cdot 2^{-(\lambda+L(\lambda))^3}$. Note that selective security is sufficient since the adversary picks the puncturing point id' before the PRF key is sampled.

[28] Remember that we have $id' = id^*$.

Finally, we construct an adversary $\mathcal{A}_0'''$ for $\mathsf{MoE}-\mathsf{MultChal}$ with security parameter $L(\lambda)+\lambda$ as follows. It simulates $\mathcal{A}_0''$, but instead of answering the queries related to id' itself, it uses the coset state tuple it obtains from its challenger. Note that since we require that id' is queried at most once to win, the single copy obtained from the challenger is sufficient. $\mathcal{A}_\ell'''$ is constructed similarly, where they use the subspace descriptions and the challenge string r_ℓ submitted to them by the challenger. This simulates the game above perfectly, since we place the coset tuple obtained from the challenger in place of the coset tuple associated with id^*, which has the same distribution since z is random. Also note that the adversary outputs the correct vectors for this coset tuple, since its choice is id^*. Therefore $\mathcal{A}'''$ wins $\mathsf{MoE}-\mathsf{MultChal}$ with probability $2^{-\lambda-L(\lambda)}-2\cdot 2^{-(\lambda+L(\lambda))^3} > 2^{-2\cdot(\lambda+L(\lambda))} > 2^{-(\lambda+L(\lambda))^2}$, which is a contradiction (Theorem 23). □

6 Identity-Based and Functional Encryption with Puncturable Master Secret Key

In this section, we give definitions for public-key identity-based encryption, along with its variant where we can puncture the master secret key [CZDC19]. We also define functional encryption whose master secret key can be punctured simultaneously at all functions such that $f(m_0) \neq f(m_1)$. Then, we show how to construct such schemes in the plain model.

6.1 Definitions

We first give the definition of usual public-key identity-based encryption.

Definition 15. *An identity-based encryption scheme with message space $\mathcal{M}$ and identity space $\mathcal{ID}$ consists of the following algorithms that satisfy the correctness guarantee below.*

- $\mathsf{Setup}(1^\lambda)$: *Takes a security parameter, λ; outputs a public key pk and a master secret key msk.*
- $\mathsf{KeyGen}(msk, id)$: *Takes the master secret key and an identity $id \in \mathcal{ID}$, outputs a secret key for the identity id.*
- $\mathsf{Enc}(pk, id, m)$: *Takes the public key pk, an identity id and a message $m \in \mathcal{M}$, outputs an encryption of m under the identity id.*
- $\mathsf{Dec}(sk, ct)$: *Takes a secret key and a ciphertext, outputs either a message or $\perp$.*

Correctness *For all messages $m \in \mathcal{M}$ and identities $id, \in \mathcal{ID}$, we require*

$$\Pr\left[\mathsf{IBE.Dec}(sk, ct) = m : \begin{array}{c} pk, msk \leftarrow \mathsf{IBE.Setup}(1^\lambda) \\ sk \leftarrow \mathsf{IBE.KeyGen}(msk, id) \\ ct \leftarrow \mathsf{Enc}(pk, id, m) \end{array}\right] = 1.$$

We define the following security notion for identity-based encryption.

Definition 16 (Adaptive Indistinguishability-Based Security for Identity-Based Encryption)**.** *Consider the following game between the challenger and an adversary $\mathcal{A}$.*

$\underline{\mathsf{IBE} - \mathsf{IND}(\lambda, \mathcal{A})}$

1. *The challenger runs* $(pk, msk) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$ *and then it submits* pk *to the adversary. It also initializes the set* ID*.*

2. ***Query Phase 1:*** *For multiple rounds, the adversary adaptively submits an identity string* $id \in \mathsf{ID}$*. For each query, the challenger samples* $sk \leftarrow \mathsf{IBE.KeyGen}(msk, id)$ *and submits* sk *to the adversary. It also adds* id *to* ID*.*

3. *The adversary outputs an identity* id^* *and a pair of messages* m_0, m_1*.*

4. *The challenger samples* $b \leftarrow \{0, 1\}$ *and* $ct \leftarrow \mathsf{IBE.Enc}(pk, id^*, m_b)$*. It submits* ct *to the adversary.*

5. ***Query Phase 2:*** *For multiple rounds, the adversary adaptively submits an identity string* $id \in \mathsf{ID}$*. For each query, the challenger samples* $sk \leftarrow \mathsf{IBE.KeyGen}(msk, id)$ *and submits* sk *to the adversary. It also adds* id *to* ID*.*

6. *The adversary outputs a guess* $b' \in \{0, 1\}$*.*

7. *The challenger outputs* 1 *if and only if* $b' = b$ *and* $id^* \notin \mathsf{ID}$*.*

We say that an identity-based encryption scheme IBE *satisfies* adaptive indistinguishability-based security *if for any QPT adversary* $\mathcal{A}$

$$\Pr[\mathsf{IBE} - \mathsf{IND}(\lambda, \mathcal{A}) = 1] \leq 1/2 + \mathsf{negl}(\lambda).$$

If the adversary outputs id^* *before* $\mathsf{IBE.Setup}$ *is run, we call it* selective indistinguishability-based security*.*

We can also define an even weaker variant where the adversary cannot query for specific identities, but is only given the keys for randomly sampled identities. While this weaker variant would be sufficient for our unclonable PKE construction (Section 7.2), since we do not know of any simpler constructions (compared to the selectively secure construction below), we do not pursue this further.

We now move onto identity-based encryption with puncturable master secret keys, introduced by Chen, Zhang, Deng, Chang [CZDC19]. This is defined to be an identity-based encryption scheme where the master secret key can be punctured at an identity so that the resulting key can be used to issue secret keys for any identity except for the punctured identity. [CZDC19] also give a construction based on hierarchical identity-based encryption.

Our definition is simpler than that of [CZDC19], which allows the adversary to adaptively query for different secret keys before selecting the identity at which the master secret key will be punctured. Our construction below can also be made secure with respect to their definition by employing an adaptively secure puncturable PRF, however, our simplified definition suffices for our unclonable primitive constructions.

Definition 17. *Identity-based encryption with puncturable master secret key is an identity-based encryption scheme (Definition 15) with the following additional algorithms and correctness guarantees.*

- $\mathsf{Punc}(msk, id)$*: Takes as input the master secret key* msk *and an identity* id*, outputs a master secret key that is punctured at* id*.*

Punctured Key Correctness *For all messages $m \in \mathcal{M}$ and identities $id, id' \in \mathcal{ID}$ such that $id \neq id'$,*

$$\Pr\left[\mathsf{IBE.Dec}(sk, ct) = m : \begin{array}{c} pk, msk \leftarrow \mathsf{IBE.Setup}(1^\lambda) \\ msk' \leftarrow \mathsf{IBE.Punc}(msk, id') \\ sk \leftarrow \mathsf{IBE.KeyGen}(msk', id) \\ ct \leftarrow \mathsf{IBE.Enc}(pk, id, m) \end{array}\right] = 1.$$

We also define a stronger version of punctured key correctness, where we require that there be no difference between sampling a secret key for an identity using the actual master secret key versus using a punctured master secret key.

Definition 18 (Strong Punctured Key Correctness)**.** *For all identities $id, id' \in \mathcal{ID}$ such that $id \neq id'$, we require*

$$(psk, msk, pk) \equiv (sk, msk, pk)$$

where

$$\begin{aligned} pk, msk &\leftarrow \mathsf{IBE.Setup}(1^\lambda) \\ sk &\leftarrow \mathsf{IBE.KeyGen}(msk, id) \\ msk' &\leftarrow \mathsf{IBE.Punc}(msk, id') \\ psk &\leftarrow \mathsf{IBE.KeyGen}(msk', id). \end{aligned}$$

Definition 19 (Puncturable Master Secret Key Security for Identity-Based Encryption)**.** *Consider the following game between the challenger and an adversary $\mathcal{A}$.*

$\underline{\mathsf{PUN} - \mathsf{IBE} - \mathsf{IND}(\lambda, \mathcal{A})}$

1. *The adversary outputs an identity id^*.*
2. *The challenger runs $(pk, msk) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$ and then $msk^* \leftarrow \mathsf{IBE.Punc}(msk, id^*)$. Then, it submits pk, msk^* to the adversary.*
3. *The adversary outputs a pair of messages m_0, m_1.*
4. *The challenger samples a challenge bit $b \leftarrow \{0, 1\}$ and computes $ct \leftarrow \mathsf{IBE.Enc}(pk, id^*, m_b)$. Then, it submits ct to the adversary.*
5. *The adversary outputs a guess $b' \in \{0, 1\}$.*
6. *The challenger outputs 1 if and only if $b' = b$.*

We say that an identity-based encryption scheme IBE *satisfies puncturable master secret key security if any QPT adversary $\mathcal{A}$,*

$$\Pr[\mathsf{PUN} - \mathsf{IBE} - \mathsf{IND}(\lambda, \mathcal{A}) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

It is easy to see that puncturable master secret key security with strong punctured key correctness implies indistinguishability-based security. We formalize this below.

Theorem 25. *Let* IBE *be an identity-based encryption scheme that satisfies [polynomial, subexponential] puncturable master secret key security (Definition 19). Then, it also satisfies [polynomial, subexponential] selective indistinguishability-based security (Definition 15).*

Proof. Suppose for a contradiction that there exist a QPT adversary $\mathcal{A}$ that wins the selective indistinguishability-based security game against IBE with probability $\varepsilon(\lambda)$. We claim that the adversary $\mathcal{A}'$ described below wins the puncturable master secret key security game with $\varepsilon(\lambda)$.

$\mathcal{A}'$ runs $\mathcal{A}$ to obtain id^* and outputs it. Then, it receives pk, msk^* from the challenger. Then, $\mathcal{A}'$ simulates the first query phase as follows. It runs $\mathcal{A}$, and whenever it queries for an identity string id, $\mathcal{A}'$ computes $sk \leftarrow \mathsf{IBE.KeyGen}(msk^*, id)$ and gives sk to $\mathcal{A}$. When $\mathcal{A}$ yields m_0, m_1, then $\mathcal{A}$ outputs these values. $\mathcal{A}'$ simulates the second query phase similarly using msk^* after receiving the challenge ciphertext from the challenger. Finally, when $\mathcal{A}$ outputs it guess b', the adversary $\mathcal{A}'$ also outputs it.

Since $id^* \notin \mathsf{ID}$, i.e., since the adversary $\mathcal{A}$ never queries for id^*, hence by strong punctured key correctness of IBE, there is no difference between sampling the secret keys using the punctured master secret key (as above) or the actual master secret key (as in the original selective indistinguishability-based security game). Hence, $\mathcal{A}'$ and the challenger above perfectly simulate the selective indistinguishability-based security game played by $\mathcal{A}$. Therefore, $\mathcal{A}'$ wins the puncturable master secret key game with probability $\varepsilon(\lambda)$.

Plugging in $\varepsilon(\lambda) = 1/\mathsf{poly}(\lambda)$ or $\varepsilon(\lambda) > \mathsf{subexp}(\lambda)$ completes the proof. □

Finally, we define functional encryption where the master secret key can be punctured at all functions f such that $f(m_0) = f(m_1)$. Note that previous works [BV18, YAL$^+$19, KNT22] define their own versions of puncturable functional encryption which is different than ours, see Section 2.6. However, throughout the paper, we will write *puncturable functional encryption* to mean our definition.

Definition 20 (Puncturable Functional Encryption)**.** *Puncturable functional encryption is a functional encryption scheme (Definition 6) with the following additional algorithms and correctness guarantees.*

- $\mathsf{Punc}(msk, m_0, m_1)$*: Takes as input the master secret key msk and outputs a master secret key that is* punctured*.*

Punctured Key Correctness *For all messages $m, m_0, m_1 \in \mathcal{M}$ and functions $f \in \mathfrak{F}$ such that $f(m_0) \neq f(m_1)$,*

$$\Pr\left[\mathsf{FE.Dec}(sk, ct) = f(m) : \begin{array}{c} pk, msk \leftarrow \mathsf{FE.Setup}(1^\lambda) \\ msk' \leftarrow \mathsf{FE.Punc}(msk, m_0, m_1) \\ sk \leftarrow \mathsf{FE.KeyGen}(msk', f) \\ ct \leftarrow \mathsf{FE.Enc}(pk, m) \end{array}\right] = 1.$$

Similar to IBE, we also define a stronger version of punctured key correctness.

Definition 21 (Strong Punctured Key Correctness)**.** *For all messages $m_0, m_1 \in \mathcal{M}$ and functions $f \in \mathfrak{F}$ such that $f(m_0) \neq f(m_1)$, we require*

$$(psk, msk, pk) \equiv (sk, msk, pk)$$

where

$$\begin{aligned} pk, msk &\leftarrow \mathsf{FE.Setup}(1^\lambda) \\ sk &\leftarrow \mathsf{FE.KeyGen}(msk, f) \\ msk' &\leftarrow \mathsf{FE.Punc}(msk, m_0, m_1) \\ psk &\leftarrow \mathsf{FE.KeyGen}(msk', f). \end{aligned}$$

Definition 22 (Puncturable Functional Encryption Security)**.** *Consider the following game between the challenger and an adversary $\mathcal{A}$.*

$\underline{\mathsf{PUN-FE-IND}(\lambda, \mathcal{A})}$

1. *Challenger samples the keys $msk, pk \leftarrow \mathsf{FE.Setup}(1^\lambda)$.*
2. *The adversary receives pk. It makes polynomially many queries by sending a function $f \in \mathcal{F}$ and receiving the corresponding functional key $sk_f \leftarrow \mathsf{FE.KeyGen}(msk, f)$.*
3. *The adversary outputs challenge messages m_0, m_1.*
4. *The challenger checks if $f(m_0) = f(m_1)$ for all $f \in \mathfrak{F}$ that was queried by the adversary. If this condition is not satisfied, it outputs 0 and terminates.*
5. *The challenger samples $msk' \leftarrow \mathsf{FE.Punc}(msk, m_0, m_1)$.*
6. *The challenger samples a challenge bit $b \leftarrow \{0, 1\}$ and prepares $ct \leftarrow \mathsf{Enc}(pk, m_b)$.*
7. *The adversary receives msk', ct and outputs a guess b'.*
8. *The challenger outputs 1 if $b' = b$.*

We say that an functional encryption scheme FE *satisfies puncturable master secret key security if any QPT adversary $\mathcal{A}$,*

$$\Pr[\mathsf{PUN-FE-IND}(\lambda, \mathcal{A}) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

6.2 Puncturable Identity-Based Encryption Construction

In this section, we show how to construct an identity-based encryption with puncturable master secret key from indistinguishability obfuscation and public-key encryption, which in turn can be constructed from indistinguishability obfuscation and one-way functions [SW14, Zha12a].

We obtain our construction through a small addition to the PKE-to-IBE transformation of [CZDC19]. In their construction of an IBE scheme, the master secret key is a (puncturable) PRF key that is used to spin up a fresh instance of a PKE scheme for each identity value. In our puncturable IBE scheme, the puncturing algorithm is simply the puncturing algorithm for the PRF family. We present the full scheme below for completeness.

Assume the existence of following schemes.

- $i\mathcal{O}$, an indistinguishability obfuscation scheme,
- PKE, a public-key encryption scheme with message space $\mathcal{M}$,
- F, a puncturable PRF family with input space $\mathcal{ID}$ and output length same as the size of the randomness used by $\mathsf{PKE.KeyGen}$,

$\underline{\mathsf{IBE.Setup}(1^\lambda)}$

1. Sample a PRF key $K \leftarrow F.\mathsf{KeyGen}(1^\lambda)$.
2. Sample $\mathsf{OPKeyGen} \leftarrow i\mathcal{O}(\mathsf{PKeyGen}_K, 1^\lambda)$, where $\mathsf{PKeyGen}_K$ is the following program.

```
$\mathsf{PKeyGen}_K(id)$
Hardcoded: $K$
    1. Sample $ipk, isk \leftarrow \mathsf{PKE.KeyGen}(1^\lambda; F(K, id))$.
    2. Output $ipk$.
```

3. Output $(\mathsf{OPKeyGen}, K)$.

```
$\mathsf{IBE.KeyGen}(msk, id)$
    1. Parse $K = msk$.
    2. Compute $(ipk, isk) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda; F(K, id))$.
    3. Output $isk$.
```

```
$\mathsf{IBE.Punc}(msk, id)$
    1. Parse $K = msk$.
    2. Compute $K' \leftarrow F.\mathsf{Punc}(K, id)$.
    3. Output $K'$.
```

```
$\mathsf{IBE.Enc}(pk, id, m)$
    1. Parse $\mathsf{OPKeyGen} = pk$.
    2. Compute $ipk \leftarrow \mathsf{OPKeyGen}(id)$.
    3. Output $\mathsf{PKE.Enc}(ipk, m)$.
```

$\mathsf{IBE.Dec}(sk, ct)$ Same as $\mathsf{PKE.Dec}$.

Theorem 26. IBE *satisfies both correctness (Definition 18) and strong punctured master secret key correctness (Definition 19).*

Proof. Correctness is easy to see, by correctness of PKE and IBE.

We move onto strong punctured master secret key correctness. Consider any $id \neq id'$. By punctured key correctness of F, we have that $F(K\{id'\}, id) = F(K, id)$ with probability 1 over the choice of the key and sampling of the punctured key. The result follows. □

Theorem 27. IBE *satisfies puncturable master secret key security (Definition 19).*

Since public-key encryption can be based on $i\mathcal{O}$ and one-way functions [SW14, Zha12a], and puncturable PRFs can be based on one-way functions also (Theorem 6), we get the following corollary.

Corollary 3. *Assuming the existence of indistinguishability obfuscation and one-way functions, there exist an identity-based encryption scheme with puncturable master secret keys, for any message length and identity length that is polynomial in the security parameter.*

We prove Theorem 27 in Section 6.3. It is easy to see that our proof also generalizes to the subexponential security case. Hence, we get the following.

Corollary 4. *Assuming the existence of subexponentially secure indistinguishability obfuscation and one-way functions, there exist a subexponentially secure identity-based encryption scheme with puncturable master secret keys.*

6.3 Proof of Security for Puncturable IBE

In this section, we prove Theorem 27. Suppose for a contradiction that there exists a QPT adversary $\mathcal{A}$ that wins the puncturable master secret key security game $\mathsf{PUN-IBE-IND}$ (Definition 19) with non-negligible probability. We prove security through a series of hybrids, each of which is constructed by modifying the previous hybrid.

Hyb_0**:** The original game $\mathsf{PUN-IBE-IND}(\lambda, \mathcal{A})$.

Hyb_1**:** The challenger computes $z^* = F(K, id^*)$ and $K\{id^*\} \leftarrow F.\mathsf{Punc}(K, id^*)$ after the adversary has submitted id^*. Then, it computes $ipk^*, isk^* \leftarrow \mathsf{PKE.KeyGen}(1^\lambda; z^*)$. Finally, instead of sampling the public key pk as before, it now computes it as $\mathsf{OPKeyGen} \leftarrow i\mathcal{O}(\mathsf{PKeyGen}'_{K\{id^*\},ipk^*,id^*}, 1^\lambda)$, where $\mathsf{PKeyGen}'_{K\{id^*\},ipk^*,id^*}$ is the following program.

```
$\mathsf{PKeyGen}'_{K\{id^*\},ipk^*,id^*}(id)$
    Hardcoded: $K\{id^*\}, ipk^*, id^*$
    1. If $id = id^*$, output $ipk^*$ and terminate.
    2. Sample $ipk, isk \leftarrow \mathsf{PKE.KeyGen}(1^\lambda; F(K, id))$.
    3. Output $ipk$.
```

Hyb_2**:** The challenger now samples z^* uniformly at random from the output space of F instead of computing it as $z^* = F(K, id^*)$.

Claim 1. $\mathsf{Hyb}_0 \approx \mathsf{Hyb}_1$.

Proof. By correctness of the punctured PRF keys, we have that $\mathsf{PKeyGen}'_{K\{id^*\},ipk^*,id^*}$ and $\mathsf{PKeyGen}_K$ have the same functionality. The result follows by the security of $i\mathcal{O}$. □

Claim 2. $\mathsf{Hyb}_1 \approx \mathsf{Hyb}_2$.

Proof. Observe that in Hyb_1, the adversary has only access to (efficient functions of) PRF evaluations at points other than id^* and the punctured PRF key $K\{id^*\}$, rather than the full key K. Therefore, the result follows from the puncturable PRF security (Definition 1). □

By above, we get that $\mathcal{A}$ wins in Hyb_2 with non-negligible probability. We claim that the adversary $\mathcal{A}'$ described below wins the public-key encryption security game against PKE with non-negligible probability.

$\mathcal{A}'$ runs $\mathcal{A}$ to obtain id^*. Then, it samples a PRF key K for F and computes $K\{id^*\} \leftarrow F.\mathsf{Punc}(K, id^*)$. When it receives the public key pk from the challenger, $\mathcal{A}'$ computes $\mathsf{OPKeyGen} \leftarrow i\mathcal{O}(\mathsf{PKeyGen}_{K\{id^*\},pk,id^*}, 1^\lambda)$, where $\mathsf{PKeyGen}_{K\{id^*\},pk,id^*}$ is the following program.

```
$\mathsf{PKeyGen}'_{K\{id^*\},pk,id^*}(id)$
    **Hardcoded:** $K\{id^*\}, pk, id^*$

    1. If $id = id^*$, output $pk$ and terminate.
    2. Sample $ipk, isk \leftarrow \mathsf{PKE.KeyGen}(1^\lambda; F(K, id))$.
    3. Output $ipk$.
```

Then, $\mathcal{A}'$ runs $\mathcal{A}$ on $\mathsf{OPKeyGen}$ and $K\{id^*\}$ to obtain m_0, m_1, which it submits to the challenger. Finally, when $\mathcal{A}'$ obtains the challenge ciphertext from the challenger, it runs $\mathcal{A}$ on it to obtain the guess b', which it submits to the challenger.

It is easy to see that $\mathcal{A}'$ wins the public-key encryption security game with the same probability as $\mathcal{A}$ wins in Hyb_2, which is non-negligible. This is a contradiction to the security of PKE.

6.4 Puncturable Functional Encryption Construction

As an application of our puncturable IBE scheme and as a warm-up to our copy-protected functional encryption scheme, we show how to construct puncturable functional encryption. In our copy-protected functional encryption scheme (Section 8), we use a punctured master secret key in a non-black-box way to remove interaction from the post-challenge-ciphertext phase of the security game.

Now we move onto our construction, which will be similar to the delegatable functional encryption scheme of [CGJS15]. Assume the existence of following schemes and we will construct a puncturable functional encryption for the class of functions $\mathfrak{F}$ defined as all circuits that are of size at most $Q(\lambda)$, where $Q(\lambda)$ is a fixed polynomial.

- $i\mathcal{O}$, $2^{-\lambda-Q(\lambda)}$-secure indistinguishability obfuscation scheme,
- IBE, a $2^{-\lambda-Q(\lambda)}$-secure public-key identity-based encryption scheme for identity space $\{0,1\}^{Q(\lambda)}$ with puncturable master secret keys (Definition 19) and deterministic identity key generation satisfying strong punctured key correctness (Definition 18)
- F, a a $2^{-\lambda-Q(\lambda)}$-secure puncturable PRF family with input space $\{0,1\}^{Q(\lambda)}$ and output length same as the size of the randomness used by $\mathsf{IBE.Enc}$,

```
$\mathsf{FE.Setup}(1^\lambda)$

1. Sample $pk, imsk \leftarrow \mathsf{IBE.Setup}(1^\lambda)$.
2. Output $pk, (imsk, \mathsf{FULL-KEY})$.
```

```
$\mathsf{FE.KeyGen}(msk', f)$

1. Parse $(msk'', \mathsf{TYPE}) = msk'$.
2. If $\mathsf{TYPE} = \mathsf{PUNC-KEY}$, output $msk''(f)$ and terminate.
3. Sample $sk \leftarrow \mathsf{IBE.KeyGen}(msk'', f)$.
4. Output $(sk, f)$.
```

```
$\underline{\mathsf{FE.Punc}(msk, m_0, m_1)}$
  1. Parse $(imsk, \mathsf{TYPE}) = msk$. Terminate if $\mathsf{TYPE} \neq \mathsf{FULL} - \mathsf{KEY}$.
  2. Sample $\mathsf{OPKey} \leftarrow i\mathcal{O}(1^\lambda, \mathsf{PKey}_{imsk,m_0,m_1})$ where $\mathsf{PKey}_{imsk,m_0,m_1}$ is the following program.

     $\underline{\mathsf{PKey}_{imsk,m_0,m_1}(f)}$
     Hardcoded: $imsk, m_0, m_1$
       1. If $f(m_0) \neq f(m_1)$, output $\perp$ and terminate.
       2. Compute $sk = \mathsf{IBE.KeyGen}(imsk, f)$.
       3. Output $(sk, f)$.

  3. Output $\mathsf{OPKey}$.
```

```
$\underline{\mathsf{FE.Enc}(pk, m)}$
  1. Sample $K \leftarrow F.\mathsf{Setup}(1^\lambda)$.
  2. Sample $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}_{m,pk,K})$ where $\mathsf{PCt}_{m,pk,K}$ is the following program.

     $\underline{\mathsf{PCt}_{m,pk,K}(f)}$
     Hardcoded: $m, pk, K$
       1. Compute $a = f(m)$.
       2. Compute $ct = \mathsf{IBE.Enc}(pk, f, a; F(K, f))$.
       3. Output $ct$.

  3. Output $\mathsf{OPCt}$.
```

```
$\underline{\mathsf{FE.Dec}(sk, ct)}$
  1. Parse $(sk', f) = sk$.
  2. Parse $\mathsf{OPCt} = ct$.
  3. $ct' = \mathsf{OPCt}(f)$.
  4. Output $\mathsf{IBE.Dec}(sk', ct')$.
```

Theorem 28. *FE satisfies both correctness (Definition 21) and strong punctured master secret key correctness (Definition 22).*

Proof. Follows in a straightforward manner from the correctness of the underlying primitives and the fact that $\mathsf{IBE.KeyGen}$ is deterministic. □

Theorem 29. *FE satisfies puncturable master secret key security (Definition 22).*

Since we construct in Section 6.2 a puncturable IBE with the properties required by FE based on $i\mathcal{O}$ and one-way functions, we get the following corollary.

Corollary 5. *Assuming the existence of subexponentially secure indistinguishability obfuscation and one-way functions, there exist a puncturable functional encryption scheme.*

6.5 Proof of Security for Puncturable Functional Encryption

In this section, we prove Theorem 29. Our proof will be similar to security proof of the delegatable functional encryption scheme in [CGJS15]. Throughout the proof, we will interpret the functions $f \in \mathfrak{F}$, which are represented by circuits of size $Q(\lambda)$, as numbers in $\{0, 1, \dots, 2^Q - 1\}$.

Suppose for a contradiction there exists a QPT adversary $\mathcal{A}$ that wins the puncturable functional encryption game with non-negligible advantage, that is, $\Pr[\mathsf{PUN} - \mathsf{FE} - \mathsf{IND}(\lambda, \mathcal{A}) = 1] \geq 1/2 + 1/p(\lambda)$ for some polynomial $p(\cdot)$ and for infinitely many values of $\lambda > 0$. We will prove security through a series of hybrids, each of which is obtained by modifying the previous one, starting with Hyb_0.

We define Hyb_0 to be the same as the original security game $\mathsf{PUN} - \mathsf{FE} - \mathsf{IND}(\lambda, \mathcal{A})$.

Hyb_t for $t \in \{0, 1, \dots, 2^Q\}$: We now compute the challenge ciphertext (encryption of m_b) as $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}^{(t)})$.

```
$\mathsf{PCt}^{(t)}(f)$
    Hardcoded: $m_0, m_1, b, pk, K$

  1. If $f < t$, set $a = f(m_{1-b})$. Otherwise, set $a = f(m_b)$.

  2. Compute $ct = \mathsf{IBE.Enc}(pk, f, a; F(K, f))$.

  3. Output $ct$.
```

Define the event E_t to be the event that the pair of challenge messages m_0, m_1 output by the adversary $\mathcal{A}$ are such that $t(m_0) = t(m_1)$.

Lemma 4. $|(\mathsf{Hyb}_t)_{|E_t} - (\mathsf{Hyb}_t)_{|E_t}| < 2^{-\lambda - Q(\lambda)}$.

Proof. Observe the programs $\mathsf{PCt}^{(t)}$ and $\mathsf{PCt}^{(t+1)}$ differ only on input $f = t$, in which case the first program compute $a = t(m_b)$ whereas the second program computes $a = t(m_{1-b})$. However, conditioned on E_t, we have $t(m_b) = t(m_{1-b})$. Hence, the programs have the same functionality and the result follows from the security of $i\mathcal{O}$. □

We will also prove $|(\mathsf{Hyb}_t)_{|\overline{E_t}} - (\mathsf{Hyb}_t)_{|\overline{E_t}}| < 2^{-\lambda/2 - Q(\lambda)}$. This combined with the above lemma gives $|\mathsf{Hyb}_t - \mathsf{Hyb}_{t+1}| < 2^{-\lambda/2 - Q(\lambda)}$ through triangle inequality. Crucially, note that the probability of the event E_t is the same in both hybrids since we only change the way we compute the challenge ciphertext (which the adversary sees after choosing m_0, m_1).

To prove $|(\mathsf{Hyb}_t)_{|\overline{E_t}} - (\mathsf{Hyb}_t)_{|\overline{E_t}}| < 2^{-\lambda/2 - Q(\lambda)}$, we define a sequence of intermediary hybrids.

$\mathsf{Hyb}_{t,1}$ for $t \in \{0, 1, \dots, 2^Q\}$: We first compute $ct^* \leftarrow \mathsf{IBE.Enc}(pk, t, f(m_b); F(K, t))$ and $K\{t\} \leftarrow F.\mathsf{Punc}(K, t)$. Then, we now compute the challenge ciphertext as $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}^{(t,1)})$.

```
$\mathsf{PCt}^{(t,1)}(f)$
    Hardcoded: $ct^*, K\{t\}, m_0, m_1, b, pk$

  1. If $f = t$, output $ct^*$ and terminate.

  2. If $f < t$, set $a = f(m_{1-b})$. Otherwise, set $a = f(m_b)$.
```

```
    3. Compute $ct = \mathsf{IBE.Enc}(pk, f, a; F(K\{t\}, f))$.
    4. Output $ct$.
```

$\underline{\mathsf{Hyb}_{t,2}\ \textbf{for}\ t \in \{0,1,\ldots,2^Q\}}$**:** We now sample $ct^* \leftarrow \mathsf{IBE.Enc}(pk, t, f(m_b); z)$ where z is sampled uniformly at random from the output space of F.

$\underline{\mathsf{Hyb}_{t,3}\ \textbf{for}\ t \in \{0,1,\ldots,2^Q\}}$: We now change the way we sample punctured master secret key as follows. At the beginning of the game, we compute $imsk' \leftarrow \mathsf{IBE.Punc}(imsk, t)$. We now output $\mathsf{OPKey} \leftarrow \mathsf{PKey}^{(t)}$.

```
$\underline{\mathsf{PKey}^{(t)}(f)}$
  Hardcoded: $imsk', m_0, m_1$
    1. If $f(m_0) \neq f(m_1)$, output $\perp$ and terminate.
    2. Compute $sk = \mathsf{IBE.KeyGen}(imsk', f)$.
    3. Output $(sk, f)$.
```

$\underline{\mathsf{Hyb}_{t,4}\ \textbf{for}\ t \in \{0,1,\ldots,2^Q\}}$: Same as $\mathsf{Hyb}_{t,3}$ but we now compute ct^* as $ct^* \leftarrow \mathsf{IBE.Enc}(pk, t, f(m_{1-b}); z)$.

$\underline{\mathsf{Hyb}_{t,5}\ \textbf{for}\ t \in \{0,1,\ldots,2^Q\}}$: Same as $\mathsf{Hyb}_{t,2}$ but we compute ct^* as $ct^* \leftarrow \mathsf{IBE.Enc}(pk, t, f(m_{1-b}); F(K,t))$.

$\underline{\mathsf{Hyb}_{t,6}\ \textbf{for}\ t \in \{0,1,\ldots,2^Q\}}$: Same as $\mathsf{Hyb}_{t,1}$ but we compute ct^* as $ct^* \leftarrow \mathsf{IBE.Enc}(pk, t, f(m_{1-b}); F(K,t))$.

Lemma 5. *$|(\mathsf{Hyb}_t)_{|\overline{E_t}} - (\mathsf{Hyb}_{t,1})_{|\overline{E_t}}| < 2^{-\lambda-Q(\lambda)}$ for all $t \in \{0,1,\ldots,2^Q\}$.*

Proof. By the punctured key correctness of F, the programs $\mathsf{PCt}^{(t)}$ and $\mathsf{PCt}^{(t,1)}$ have the same functionality. The results follows by the security of $i\mathcal{O}$. □

Lemma 6. *$|(\mathsf{Hyb}_{t,1})_{|\overline{E_t}} - (\mathsf{Hyb}_{t,2})_{|\overline{E_t}}| < 2^{-\lambda-Q(\lambda)}$ for all $t \in \{0,1,\ldots,2^Q\}$.*

Proof. Observe that the adversary only has the punctured key $K\{t\}$. The resut follows by the security of the puncturable PRF F. □

Lemma 7. *$|(\mathsf{Hyb}_{t,2})_{|\overline{E_t}} - (\mathsf{Hyb}_{t,3})_{|\overline{E_t}}| < 2^{-\lambda-Q(\lambda)} \cdot \mathsf{poly}(\lambda)$ for all $t \in \{0,1,\ldots,2^Q\}$.*

Proof. Since we are conditioned on the event $\overline{E_t}$, we have $t(m_0) \neq t(m_1)$. Hence, by strong punctured key correctness of IBE, all the programs PKey_P in these hybrids have the same functionality. Result follows from the security of $i\mathcal{O}$. □

Lemma 8. *$|(\mathsf{Hyb}_{t,3})_{|\overline{E_t}} - (\mathsf{Hyb}_{t,4})_{|\overline{E_t}}| < 2^{-\lambda-Q(\lambda)}$ for all $t \in \{0,1,\ldots,2^Q\}$.*

Proof. Since we are conditioned on the event $\overline{E_t}$, we have $t(m_0) \neq t(m_1)$. Further, to win, for any function $f \in \mathfrak{F}$, if f is queried by the adversary, then $f(m_0) = f(m_1)$. Combining these, we get that t was never queried by the adversary. Therefore, the adversary never gets a secret key for the identity t. In particular, all the identity secret keys obtained by the adversary (as a result of functional key queries) can instead be obtained using $imsk'$, the IBE master secret key punctured at t, due to the strong punctured key correctness of IBE. Further, the punctured IBE master secret key obtained by the adversary (which is inside the punctured FE master secret key) is also punctured at t. Finally, observe that ct^* is an IBE encryption (sampled using true randomness z) under the identity t. Hence, the result follows by puncturable IBE security (Definition 19). □

Lemma 9. $|(\mathsf{Hyb}_{t,4})_{|\overline{E_t}} - (\mathsf{Hyb}_{t,5})_{|\overline{E_t}}| < 2^{-\lambda - Q(\lambda)} \cdot \mathsf{poly}(\lambda)$ *for all* $t \in \{0, 1, \ldots, 2^Q\}$.

Proof. Same argument as Lemma 7. □

Lemma 10. $|(\mathsf{Hyb}_{t,5})_{|\overline{E_t}} - (\mathsf{Hyb}_{t,6})_{|\overline{E_t}}| < 2^{-\lambda - Q(\lambda)}$ *for all* $t \in \{0, 1, \ldots, 2^Q\}$.

Proof. Same argument as Lemma 6. □

Lemma 11. $|(\mathsf{Hyb}_{t,6})_{|\overline{E_t}} - (\mathsf{Hyb}_{t+1})_{|\overline{E_t}}| < 2^{-\lambda - Q(\lambda)}$ *for all* $t \in \{0, 1, \ldots, 2^Q\}$.

Proof. Same argument as Lemma 5. □

Combining the above lemmata, we get $|(\mathsf{Hyb}_t)_{|\overline{E_t}} - (\mathsf{Hyb}_t)_{|\overline{E_t}}| < 2^{-\lambda/2 - Q(\lambda)}$. Then, as argued before, this gives $|\mathsf{Hyb}_t - \mathsf{Hyb}_{t+1}| < 2^{2^{-\lambda/2 - Q(\lambda)}}$, finally yielding $|\mathsf{Hyb}_0 - \mathsf{Hyb}_{2^Q}| < 2^{-\lambda/2}$. Hence, $\Pr[\mathsf{Hyb}_0 = 1] \geq 1/2 + 1/p(\lambda)$ by assumption and therefore $\Pr[\mathsf{Hyb}_{2^Q} = 1] \geq 1/2 + 1/(2 \cdot p(\lambda))$. However, this is clearly a contradiction, since the challenge messages in these hybrids are m_b and m_{1-b} respectively (while we still compare the adversary's guess to b).

7 Public-Key Encryption with Copy-Protected Secret Keys

In this section, we define public-key encryption with copy-protected secret keys. Then, we give our construction based on coset states and prove it secure.

7.1 Definitions

Definition 23 (Public-key Encryption with Copy-Protected Secret Keys)**.** *A public-key encryption scheme with copy-protected secret keys consists of the following efficient algorithms.*

- $\mathsf{KeyGen}(1^\lambda)$*: Takes in the security parameter, output a classical secret key* sk *and a public key* pk*.*
- $\mathsf{QKeyGen}(sk)$*: Takes as input the classical secret key and outputs a quantum secret key.*
- $\mathsf{Enc}(pk, m)$*: Takes in the public key and a message* $m \in \mathcal{M}$*, outputs and encryption of* m*.*
- $\mathsf{Dec}(\mathsf{R}_{\mathsf{dec}}, ct)$*: Takes in a quantum secret key and a ciphertext, outputs a message or* $\perp$*.*

We require correctness[29] *and* CPA security*.*

Correctness *For all messages* $m \in \mathcal{M}$*,*

$$\Pr\left[\mathsf{Dec}(\mathsf{R}_{\mathsf{dec}}, ct) = m : \begin{array}{c} pk, sk \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{R}_{\mathsf{dec}} \leftarrow \mathsf{QKeyGen}(sk) \\ ct \leftarrow \mathsf{Enc}(pk, m) \end{array}\right] = 1.$$

[29] While our schemes satisfy perfect correctness, i.e., correctness with probability 1, some work relax the definition to $1 - \mathsf{negl}(\lambda)$.

CPA Security *For any stateful QPT adversary $\mathcal{A}$,*

$$\Pr\left[\mathcal{A}(ct) = b : \begin{array}{c} pk, sk \leftarrow \mathsf{Setup}(1^\lambda) \\ m_0, m_1 \leftarrow \mathcal{A}(pk, 1^\lambda) \\ b \leftarrow \{0,1\} \\ ct \leftarrow \mathsf{Enc}(pk, m_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

As observed by [CLLZ21], correctness of the scheme along with Almost As Good As New Lemma (Lemma 1) means that we can implement decryption in a way such that the quantum secret key is not disturbed. Thus, we can reuse the key to decrypt any number of times.

Following prior work, we will use two different security notions, regular anti-piracy and strong anti-piracy. The former will be the natural security notion while the latter definition is easier to work with when proving security. Both of our definitions follow [CLLZ21, LLQZ22], with the strengthening that we allow unbounded[30] number of key queries and we also allow the adversary to choose different challenge messages for each freeloader.

Now, we move onto the first definition. In this definition, the pirate (or *splitting*) adversary queries for copy-protected keys for any number of rounds. Then, if it has queried for k keys, it outputs $k+1$ freeloaders, which are unitaries along with hardwired quantum states. More precisely, it outputs a $(k+1)$-partite (possibly entangled) register $\mathsf{R}_{\mathsf{adv}}$ and unitaries U_ℓ. Then, the challenger presents these freeloaders with challenge ciphertexts, and the adversary wins if all freeloaders correctly predict the challenges. Below, we write $\mathsf{U}_{quantum}$ to denote the quantum universal circuit $\mathsf{U}_{quantum}((U, \rho), x)$ that takes in a unitary U and a state ρ, and simulates the *induced* quantum circuit on input x (i.e. computes $U(\rho, x)$), and finally measures the first output qubit in the computational basis. We note that the freeloaders being unitaries is not restrictive and actually captures general quantum circuits since the hardwired quantum state $(\mathsf{R}_{\mathsf{adv}})_\ell$ can include[31] workspace qubits initialized to zeroes.

Definition 24 (CPA-Style Regular γ-Anti-Piracy Security)**.** *Let* PKE *be a public key encryption scheme with copy-protected secret keys. Consider the following game between the challenger and an adversary $\mathcal{A}$.*

$\underline{\mathsf{PKEAntiPiracy}(\lambda, \mathcal{A})}$

1. *The challenger runs $sk, pk \leftarrow$ $\mathsf{PKE.Setup}(1^\lambda)$ and submits pk to the adversary.*
2. *For multiple rounds, $\mathcal{A}$ makes quantum key queries. For each query, the challenger generates a key as $\mathsf{R} \leftarrow \mathsf{PKE.QKeyGen}(sk)$ and submits R to the adversary.*
3. *$\mathcal{A}$ outputs a $(k+1)$-partite register $\mathsf{R}_{\mathsf{adv}}$, unitaries $\{U_\ell\}_{\ell \in [k+1]}$ and challenge messages $\{m_\ell^0, m_\ell^1\}_{\ell \in [k+1]}$, where k is the number of queries it made.*
4. *The challenger executes the following for each $\ell \in [k+1]$.*

   4.1. $b_\ell \leftarrow \{0,1\}$.

   4.2. $ct_\ell \leftarrow \mathsf{PKE.Enc}(pk, m_\ell^{b_\ell})$.

   4.3. $b'_\ell \leftarrow \mathsf{U}_{quantum}(U_\ell, \mathsf{R}_{\mathsf{adv}}[\ell], ct_\ell)$.

[30] Still polynomial since the adversary is QPT.

[31] It will also include some quantum information that the pirate adversary has produced from the copy-protected keys.

4.4. Check if $b'_\ell = b_\ell$.

5. *The challenger outputs 1 if and only if all the checks pass.*

We say that PKE *satisfies γ-anti-piracy security if for any QPT adversary $\mathcal{A}$,*

$$\Pr[\mathsf{PKEAntiPiracy}(\lambda, \mathcal{A}) = 1] \leq \frac{1}{2} + \gamma(\lambda) + \mathsf{negl}(\lambda).$$

We ignore writing γ when $\gamma = 0$.

Note that we can also define a version where the freeloader adversaries try to guess the whole message m_ℓ where $m_\ell \leftarrow \mathcal{M}$; and we require negligible probability of success. It is not known if this version is not implied by CPA security, see [CLLZ21]. However, our construction will satisfy both notions. Before moving onto the stronger definition, we need the following notation.

Definition 25 (Decryptor Testing)**.** *In the anti-piracy game between the challenger and an adversary, fix $\ell \in [k+1]$, some values m_ℓ^0, m_ℓ^1 of the challenge messages, a freeloader unitary U_ℓ, and some value st of a classical state maintained by the challenger (which will be defined later). Let $\mathcal{D}$ be an efficient ciphertext distribution that can depend on st. That is, $\mathcal{D}^{st}(m; r)$ is an efficient classical algorithm where $m \in \mathcal{M}$, $r \in \mathcal{R}$ and $\mathcal{R}$ is a random coin set.*

Consider the following mixture $\mathcal{P}$ of binary projective measurements, induced by $\mathcal{D}$ and $m_\ell^0, m_\ell^1, U_\ell, st$, applied on a state ρ.

1. *Sample $b \leftarrow \{0, 1\}$.*

2. *Sample $r \leftarrow \mathcal{R}$.*

3. *Run $ct \leftarrow \mathcal{D}^{st}(m_\ell^b; r)$.*

4. *Execute U_ℓ on (ρ, ct), and measure the first qubit of the output registers, let b' be the output.*

5. *Output 1 if $b' = b$. Otherwise, output 0.*

Observe that we can efficiently execute the above measurement[32] *for arbitrary given superpositions of r and b values. Therefore, by Section 4, there exists both exact and approximated projective and threshold implementations for $\mathcal{P}$. We write $\mathsf{PI}_{\ell,\mathcal{D}}$ and $\mathsf{API}_{\ell,\mathcal{D}}^{\varepsilon,\delta}$ to denote the projective implementation and approximate projective implementation of $\mathcal{P}$, respectively. Similarly, let $\mathsf{TI}_{\ell,\mathcal{D},\eta}$ and $\mathsf{ATI}_{\ell,\mathcal{D},\eta}^{\varepsilon,\delta}$ denote the threshold and efficient approximate threshold implementations of $\mathcal{P}$ for a threshold value η.*

While the fixed values $m_\ell^0, m_\ell^1, U_\ell, st$ are omitted from the notation, they will be clear from the context. Unless otherwise specified, we will write $\mathcal{D}$ to denote the honest ciphertext distribution, that is, we encrypt m as

$$ct \leftarrow \mathsf{PKE.Enc}(pk, m)$$

where pk is part of st.

Definition 26 (CPA-Style Strong γ-Anti-Piracy)**.** *Let* PKE *be a public key encryption scheme with copy-protected secret keys. Consider the following game between the challenger and an adversary $\mathcal{A}$.*

[32] More formally, we are actually talking about the measurement where r, b are fixed

$\underline{\mathsf{PKEStrongAntiPiracy}(\lambda, \gamma(\lambda), \mathcal{A})}$

1. *The challenger runs* $sk, pk \leftarrow \mathsf{PKE.Setup}(1^\lambda)$ *and submits* pk *to the adversary.*

2. *For multiple rounds,* $\mathcal{A}$ *makes quantum key queries. For each query, the challenger generates a key as* $\mathsf{R} \leftarrow \mathsf{PKE.QKeyGen}(sk)$ *and submits* R *to the adversary.*

3. $\mathcal{A}$ *outputs a* $(k+1)$*-partite register* $\mathsf{R}_{\mathsf{adv}}$*, unitaries* $\{U_\ell\}_{\ell \in [k+1]}$ *and challenge messages* $\{m_\ell^0, m_\ell^1\}_{\ell \in [k+1]}$*, where* k *is the number of queries it made.*

4. *The challenger applies the test*
$$\bigotimes_{\ell \in [k+1]} \mathsf{TI}_{\ell, \mathcal{D}, 1/2+\gamma}$$
to $\mathsf{R}_{\mathsf{adv}}$ *and outputs* 1 *if and only if the measurement result is all* 1*.*

We say that PKE *satisfies strong* γ*-anti-piracy security if for any QPT adversary* $\mathcal{A}$*,*

$$\Pr[\mathsf{PKEStrongAntiPiracy}(\lambda, \gamma(\lambda), \mathcal{A})] \leq \mathsf{negl}(\lambda).$$

We also have the following relationship between the various security definitions for public-key encryption.

Theorem 30 ([CLLZ21])**.** *Suppose a public key encryption scheme with copy-protected keys satisfies CPA-style strong* γ*-anti-piracy (Definition 26). Then, it also satisfies CPA-style regular* γ*-anti-piracy (Definition 24).*

While [CLLZ21] proves the above for only $1 \to 2$ anti-piracy security, it can be generalized to unbounded collusion setting - see proof of Theorem 34.

Theorem 31 ([CLLZ21])**.** *Suppose a public key encryption scheme with copy-protected keys satisfies CPA-style regular* γ*-anti-piracy (Definition 24) for any inverse polynomial* γ*. Then, it also satisfies regular CPA security and regular* γ*-anti-piracy for* $\gamma = 0$*.*

This is simply due to the definition of $\mathsf{negl}(\lambda)$ and γ-anti-piracy.

7.2 Construction

In this section, we present our construction. Assume the existence of following primitives where we set $\nu(\lambda) = 2^{-6\lambda} \cdot 2^{-8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$.

- $i\mathcal{O}$, indistinguishability obfuscation scheme that is $\nu(\lambda)$-secure against $2^{5\lambda} \cdot 2^{8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$-time adversaries,

- IBE, identity-based encryption scheme for the identity space $\mathcal{ID} = \{0,1\}^\lambda$ (Definition 15) that is $\nu(\lambda)$-secure against $2^{5\lambda} \cdot 2^{8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$-time adversaries,

- F_1, puncturable PRF family with input length λ and output length same as the size of the randomness used by $\mathsf{CosetGen}$ (Definition 14) that is $\nu(\lambda)$-secure against $2^{5\lambda} \cdot 2^{8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$-time adversaries,

- F_2, puncturable PRF family with input length λ and output length same as the size of the randomness used by $\mathsf{IBE.Enc}$ that is $\nu(\lambda)$-secure against $2^{5\lambda} \cdot 2^{8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$-time adversaries,

- CCObf, compute-and-compare obfuscation for $2^{-\lambda^{0.2\cdot C_{\mathsf{MoE.Coll}}}}$-unpredictable distributions that is $2^{-2\lambda-1}\cdot 2^{-2\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$-secure against $2^{3\lambda}\cdot 2^{2\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$-time adversaries,

A remark is in order regarding our assumptions. We note that all of our assumptions above can be based on any subexponential $i\mathcal{O}$ and LWE assumption. For example, if we have an $i\mathcal{O}$ scheme that is $2^{-\lambda^{c_1}}$-secure against $2^{\lambda^{c_2}}$-time adversaries; in our construction we implicitly initiate it with security parameter $\lambda^{c'}$ where $c' = \max\{0.3C_{\mathsf{MoE.Coll}}/c_1, 0.3C_{\mathsf{MoE.Coll}}/c_2\}$. While this might require larger padding for obfuscated circuits, this is still within polynomial factors. The same applies for the other primitives. Thus, our assumptions can be based solely on subexponential hardness for any exponent, since we can always scale the security parameter by a polynomial factor when instantiating the underlying primitives.

Set $L(\lambda) = \lambda$ and therefore $c_L(\lambda) = 24\cdot\lambda^3$ (see Theorem 24). We also assume that all obfuscated programs in the construction and in the proof are appropriately padded.

We now give our construction for public-key encryption with copy-protected secret keys.

$\underline{\mathsf{PKE.Setup}(1^\lambda)}$

```
1. Sample a PRF key $K_1 \leftarrow F_1.\mathsf{KeyGen}(1^\lambda)$.
2. Sample $cpk, csmk \leftarrow \mathsf{IBE.Setup}(1^\lambda)$.
3. Sample $\mathsf{OPMem} \leftarrow i\mathcal{O}(\mathsf{PMem}_{K_1})$, where $\mathsf{PMem}_{K_1}$ is the following program.

    $\mathsf{PMem}_{K_1}(id, u_1, \ldots, u_{c_L(\lambda)}, r)$
    Hardcoded: $K_1$
    1. $(A_i, s_i, s'_i)_{i\in[c_L(\lambda)]} \leftarrow \mathsf{CosetGen}(1^{L(\lambda)+\lambda}; F_1(K_1, id))$.
    2. For each $i \in [c_L(\lambda)]$, check if $u_i \in A_i + s_i$ if $(r)_i = 0$ and check if $u_i \in A_i^\perp + s'_i$ if
       $(r)_i = 1$. If any of the checks fail, output 0 and terminate.
    3. Output 1.

4. Set $pk = (cpk, \mathsf{OPMem})$ and $sk = (cmsk, K_1)$.
5. Output $(pk, sk)$.
```

$\underline{\mathsf{PKE.QKeyGen}(sk)}$

```
1. Parse $(cmsk, K_1) = sk$.
2. Sample $id \leftarrow \{0,1\}^\lambda$.
3. $(A_i, s_i, s'_i)_{i\in[c_L(\lambda)]} = \mathsf{CosetGen}(1^{L(\lambda)+\lambda}; F_1(K_1, id))$.
4. $ck \leftarrow \mathsf{IBE.KeyGen}(cmsk, id)$.
5. Output $\left(\left|A_{i,s_i,s'_i}\right\rangle\right)_{i\in[c_L(\lambda)]}, ck, id$.
```

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

1. Parse $(cpk, \mathsf{OPMem}) = pk$.
2. Sample $r \leftarrow \{0,1\}^{c_L(\lambda)}$.
3. Sample a PRF key K_2 for F_2 as $K_2 \leftarrow F_2.\mathsf{KeyGen}(1^\lambda)$.
4. Sample $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}_{\mathsf{OPMem},cpk,K_2,r,m})$, where $\mathsf{PCt}_{\mathsf{OPMem},cpk,K_2,r,m}$ is the following program.

```
$\mathsf{PCt}_{\mathsf{OPMem},cpk,K_2,r,m}(id, u_1, \ldots, u_{c_L(\lambda)})$
Hardcoded: $\mathsf{OPMem}, cpk, K_2, r, m$
  1. Run $\mathsf{OPMem}(id, u_1, \ldots, u_{c_L(\lambda)}, r)$. If it outputs 0, output $\bot$ and terminate.
  2. Output $\mathsf{IBE.Enc}(cpk, id, m; F_2(K_2, id))$.
```

5. Output (OPCt, r).

$\mathsf{PKE.Dec}(\mathsf{R}_{\mathsf{key}}, ct)$

1. Parse $((\mathsf{R}_i)_{i \in [c_L(\lambda)]}, ck, id) = \mathsf{R}_{\mathsf{key}}$ and $(\mathsf{OPCt}, r) = ct$.
2. For indices $i \in [c_L(\lambda)]$ such that $(r)_i = 1$, apply $H^{\otimes \kappa(L(\lambda)+\lambda)}$ to R_i.
3. Run the program OPCt coherently on id and $(\mathsf{R}_i)_{i \in [c_L(\lambda)]}$.
4. Measure the output register and denote the outcome by cct.
5. Output $\mathsf{IBE.Dec}(ck, cct)$.

Correctness with probability 1 follows in a straightforward manner from the correctness of the underlying schemes. We claim that the construction is also secure.

Theorem 32. PKE *satisfies strong γ-anti-piracy for any inverse polynomial γ.*

When we instantiate the assumed building blocks with known constructions, we get the following corollary.

Corollary 6. *Assuming subexponentially secure $i\mathcal{O}$ and subexponentially secure LWE, there exists a public-key encryption scheme that satisfies anti-piracy security against unbounded collusion.*

Proof. IBE can be constructed based on $i\mathcal{O}$ and one-way functions (Corollary 4). F_1 and F_2 can be constructed based on one-way functions (Theorem 6). which in turn can be obtained from LWE. CCObf can be constructed based on $i\mathcal{O}$ and LWE (Theorem 12). □

7.3 Proof of Strong Anti-Piracy Security

In this section, we will prove Theorem 32. We note that our construction also satisfies random challenge anti-piracy security; we only give the full proof for Theorem 32 and the random challenge anti-piracy security follows by mostly the same proof.

Throughout the proof, we will interpret identity strings for IBE, which are λ-bit strings, as integers in the set $\{0, 1, \ldots, 2^\lambda - 1\}$. Without loss of generality, we assume that IBE can also encrypt the symbol $\top$, which is outside the message space $\mathcal{M}$ for PKE.

Fix any inverse polynomial $\gamma(\lambda)$ and suppose for a contradiction that there exists an efficient adversary $\mathcal{A}$ that wins the strong γ-anti-piracy game with non-negligible probability. Let k denote the number of keys obtained by the adversary. Define Hyb_0 to be the original security game $\mathsf{PKEStrongAntiPiracy}(\lambda, \gamma(\lambda), \mathcal{A})$.

Making Key Identities Unique

Define Hyb_1 by modifying Hyb_0 as follows. We change the way we sample the identity strings in $\mathsf{PKE.QKeyGen}$ during each quantum key query. Let the challenger record each sampled identity when answering each query, and when answering a new query, it samples uniformly at random an identity value from the set $\{1, \ldots, 2^\lambda - 1\}$ *that has not appeared before*[33]. That is, we sample unique identity strings for each quantum key. Also, we define the following notation. Let $id_{\alpha(i)}$ be the i^{th} identity value sampled where $\alpha(\cdot)$ is the permutation $[k] \to [k]$ such that $0 < id_1 < \cdots < id_k < 2^\lambda$. That is, $id_{\alpha(i)}$ is the identity string that is sampling during the i^{th} query of the adversary. For simplicity of notation, we also set $id_0 = 0$ and $id_{k+1} = 2^\lambda$.

Claim 3. $|\mathsf{Hyb}_0 - \mathsf{Hyb}_1| < \exp(-\lambda)$.

Proof. An easy calculation shows that uniformly and independently sampling from $\{0, 1, \ldots, 2^\lambda - 1\}$ k times gives k unique values from the set $\{1, \ldots, 2^\lambda - 1\}$ with probability at least

$$1 - \frac{k^2(\lambda)}{2^\lambda}.$$

The result follows since $k(\cdot)$ is a polynomial. □

Making the Challenger Efficient

Define Hyb_2 by modifying Hyb_1 as follows. At the end of the game, instead of applying threshold implementations $\mathsf{TI}_{\ell,\mathcal{D},1/2+\gamma}$, the challenger applies approximate threshold implementations $\mathsf{ATI}^{\varepsilon,\delta}_{\ell,\mathcal{D},1/2+\frac{31\gamma}{32}}$ with $\varepsilon = \frac{\gamma}{32k}$ and $\delta = 2^{-10\lambda} \cdot 2^{-10\lambda^{C_{\mathsf{MoE.Coll}}}}$. It outputs 1 if and only if all ATI output 1.

Claim 4. $\Pr[\mathsf{Hyb}_2 = 1] > \Pr[\mathsf{Hyb}_1 = 1] - \exp(-\lambda)$.

Proof. Let σ be the $(k+1)$-partite state output by the adversary. By Theorem 18, we get

$$\mathrm{Tr}\left[\left(\bigotimes_{\ell \in [k+1]} \mathsf{ATI}^{\varepsilon,\delta}_{\ell,\mathcal{D},1/2+\frac{31\gamma}{32}}\right)\sigma\right] \geq \mathrm{Tr}\left[\left(\bigotimes_{\ell \in [k+1]} \mathsf{TI}_{\ell,\mathcal{D},1/2+\gamma}\right)\sigma\right] - (k(\lambda)+1) \cdot \exp(-\lambda).$$

Observe that the trace expressions on the left-hand side and the right-hand side are the winning probabilities in Hyb_2 and Hyb_1 respectively. The result follows since $k(\lambda)$ is a polynomial. □

[33] Note that this can be done on-the-go in polynomial time, with overwhelming probability, e.g. through rejection sampling

Therefore, $\mathcal{A}$ wins in Hyb_2 with probability $\frac{1}{p(\cdot)}$ for some polynomial $p(\cdot)$ and infinitely many values of $\lambda > 0$. Note that in Hyb_2, now the challenger is also efficient by Theorem 17 and our choice of ε, δ.

The rest of the proof will be devoted to showing that using $\mathcal{A}$, we can construct an adversary that breaks the selective monogamy-of-entanglement game $\mathsf{MoE} - \mathsf{Coll} - \mathsf{Sel}$ (Theorem 24). We will use projective and threshold implementations for various mixtures of measurements to test the freeloaders. The public key pk, the identity strings $id_1, \dots, id_k$ and the permutation α will be part of the classical state st of the challenger, in the sense of Definition 25. The particular distribution on the collection of projective measurements (induced by a challenge ciphertext distribution) will vary, and it will be denoted explicitly.

Definition 27. *For all $j \in [k]$, let $(A_i^j, s_i^j, s_i'^j)_{i\in[c_L(\lambda)]}$ denote the tuple of subspaces and vectors sampled during the sampling of the $(\alpha^{-1}(j))$-th key. That is, it is the coset tuple associated with id_j.*

A Monogamy-of-Entanglement Type Game

First, we define the following monogamy-of-entanglement-type game $\mathcal{G}$ for a tuple of adversaries $(\mathcal{A}'_0, \mathcal{A}'_1, \mathcal{A}'_2)$. Observe that it will be straightforward to reduce the game $\mathcal{G}$ to $\mathsf{MoE} - \mathsf{Coll} - \mathsf{Sel}$ with no loss of security, since the former is the same as the latter except that it includes an independent IBE instance that can sampled by the reduction.

$\underline{\mathcal{G}(\lambda, (\mathcal{A}'_0, \mathcal{A}'_1, \mathcal{A}'_2))}$

1. The adversary outputs an index $j^* \in [k]$.
2. The challenger executes $pk, sk \leftarrow \mathsf{PKE.Setup}(1^\lambda)$ and submits pk to $\mathcal{A}'_0$.
3. For k rounds, $\mathcal{A}'$ makes quantum key queries. For each query, the challenger samples a quantum key as in $\mathsf{PKE.QKeyGen}$, but by sampling the identity id in a collision-free way (as in Hyb_1), and submits it to $\mathcal{A}'_0$.
4. The adversary outputs a *bipartite* register $\mathsf{R}_{\mathsf{bip}}$.
5. For $\ell \in \{1, 2\}$, the challenger does the following.
   5.1. Sample $r_\ell \leftarrow \{0,1\}^{c_L(\lambda)}$.
   5.2. Run $\mathcal{A}'_\ell$ on $\mathsf{R}_{\mathsf{bip}}[\ell]$, $(A_i^{j^*})_{i\in[c_L(\lambda)]}$ and r_ℓ to obtain a tuple of vectors $(v_{\ell,i})_{i\in[c_L(\lambda)]}$.
   5.3. For all $i \in [c_L(\lambda)]$, check if $v_{\ell,i} \in A_i^{j^*} + s_i^{j^*}$ if $(r_\ell)_i = 0$ and check if $v_{\ell,i} \in (A^{j^*})_i^\perp + s_i'^{j^*}$ if $(r_\ell)_i = 1$.

   If all the checks pass, the challenger outputs 1. Otherwise, it outputs 0.

Now, we construct a tuple of adversaries $(\mathcal{A}'_0, \mathcal{A}'_1, \mathcal{A}'_2)$ for $\mathcal{G}$, starting with $\mathcal{A}'_0$. Let $\mathcal{D}_j$ for $j \in \{0, \dots, k+1\}$ be efficient ciphertext distributions, which we will define later.

$\underline{\mathcal{A}'_0(pk)}$

1. Uniformly at random sample x, y, j^* such that $1 \le x < y \le k+1$ and $j^* \in \{1, \dots, k\}$. Output j^*.
2. Simulate $\mathcal{A}$ on pk by making a quantum secret key query to the challenger whenever $\mathcal{A}$

makes a query, and forwarding the obtained key to it. Let $\mathsf{R}_{\mathsf{adv}}$ be the $(k+1)$-partite register (with state σ), let $(m_\ell^0, m_\ell^1)_{\ell\in[k+1]}$ be the challenge messages and let $(U_\ell)_{\ell\in[k+1]}$ be the unitaries output by $\mathcal{A}$ at the end of the query phase.

3. Apply $\mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}_0}$ to all registers $\mathsf{R}_{\mathsf{adv}}[\ell]$ for $\ell \in [k+1]$, let $b_{\ell,0}$ be the measurement outcomes.

4. Apply $\mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}_i}$ in succession for $i = 1$ to j^* to $\mathsf{R}_{\mathsf{adv}}[x]$, let $b_{x,i}$ be the measurement outcomes.

5. Apply $\mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}_i}$ in succession for $i = 1$ to j^* to $\mathsf{R}_{\mathsf{adv}}[y]$, let $b_{y,i}$ be the measurement outcomes.

6. Output

$$\begin{aligned}&((\mathsf{R}_{\mathsf{adv}}[x], j^*, x, y, (b_{\ell,0})_{\ell\in[k+1]}, (b_{x,i})_{i\in[j^*]}, (b_{y,i})_{i\in[j^*]}, (U_\ell)_{\ell\in[k+1]}),\\ &(\mathsf{R}_{\mathsf{adv}}[y], j^*, x, y, (b_{\ell,0})_{\ell\in[k+1]}, (b_{x,i})_{i\in[j^*]}, (b_{y,i})_{i\in[j^*]}, (U_\ell)_{\ell\in[k+1]}),\\ &j^*).\end{aligned}$$

For $j \in \{1, \dots, k\}$, define $\mathcal{D}_j$ to be the challenge ciphertext distribution where an encryption of a message m is computed as follows.

1. Sample $r \leftarrow \{0,1\}^{c_L(\lambda)}$.
2. Sample a PRF key K_2 for $F_2.\mathsf{KeyGen}(1^\lambda)$.
3. Sample $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}^{(j)}_{\mathsf{OPMem},cpk,K_2,r,m,id_j})$

$\mathsf{PCt}^{(j)}_{\mathsf{OPMem},cpk,K_2,r,m,id_j}(id, u_1, \dots, u_{c_L(\lambda)})$

Hardcoded: $\mathsf{OPMem}, cpk, K_2, r, m, id_j$

1. Run $\mathsf{OPMem}(id, u_1, \dots, u_{c_L(\lambda)}, r)$. If it outputs 0, output $\perp$ and terminate.
2. If $id < id_j$, set $a = \top$. Otherwise, set $a = m$.
3. Output $\mathsf{IBE.Enc}(cpk, id, a; F_2(K_2, id))$.

4. Output (OPCt, r).

We define $\mathcal{D}_0$ to be the honest ciphertext distribution $\mathcal{D}$ and we define $\mathcal{D}_{k+1}$ as follows.

1. Sample $r \leftarrow \{0,1\}^{c_L(\lambda)}$.
2. Sample a PRF key K_2 for $F_2.\mathsf{KeyGen}(1^\lambda)$.
3. Sample $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}^{(k+1)}_{\mathsf{OPMem},cpk,K_2,r})$

$\mathsf{PCt}^{(k+1)}_{\mathsf{OPMem},cpk,K_2,r}(id, u_1, \dots, u_{c_L(\lambda)})$

Hardcoded: $\mathsf{OPMem}, cpk, K_2, r$

1. Run $\mathsf{OPMem}(id, u_1, \dots, u_{c_L(\lambda)}, r)$. If it outputs 0, output $\perp$ and terminate.

2. Output $\mathsf{IBE.Enc}(cpk, id, \mathsf{T}; F_2(K_2, id))$.

4. Output (OPCt, r).

Note that the distribution $\mathcal{D}_{k+1}$ does not actually use the message m.

Observe that $\mathcal{A}_0'$ can indeed execute $\mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}_i}$. The identity strings id_j are part of the quantum secret keys. Further, the adversary can record the order in which the identity strings are received and also their sorted version, so it can indeed index them as id_j.

Finally, we claim that there exists efficient $\mathcal{A}_1', \mathcal{A}_2'$ such that $(\mathcal{A}_0', \mathcal{A}_1', \mathcal{A}_2')$ wins $\mathcal{G}$ with probability

$$\frac{1}{2^{0.4\cdot\lambda^{C_{\mathsf{MoE.Coll}}}}}.$$

We will construct these adversaries in the rest of the proof, and at the end we will show that the security of $\mathcal{G}$ can be reduced to $\mathsf{MoE}-\mathsf{Coll}-\mathsf{Sel}$, arriving at a contradiction.

Finding a Simultaneous Jump

In the rest of the proof, we will formalize the following fact. Observe that the adversary only obtains k different identity keys for IBE. Therefore, informally, by the security of IBE and by the pigeonhole principle, two of the $k+1$ freeloaders must be using IBE encryptions of their challenge message under the same identity id_j to decode their challenge ciphertext. This will in turn mean that they must be using the same coset state tuple, and therefore we will extract coset vectors for the same identity/tuple, which will be a contradiction by the monogamy-of-entanglement property.

As a first step, we will show that the decryption success probabilities of the two freeloaders x, y, when tested with respect to the distributions $\mathcal{D}_j$, jump at the same index j^*, meaning that the two freeloaders use the same identity string block $[id_{j^*}, id_{j^*+1} - 1]$ to decrypt.

Claim 5. *Let τ be the state of the bipartite register $\mathsf{R}_{\mathsf{adv}}[x, y]$ output by $\mathcal{A}_0'$ in $\mathcal{G}$, and also consider the classical values $j^*, x, y, \{b_{\ell,i}\}_{\ell,i}$ contained in the output of $\mathcal{A}_0'$.*

Suppose we apply the measurement $\mathsf{API}^{\varepsilon,\delta}_{x,\mathcal{D}_{j^*+1}} \otimes \mathsf{API}^{\varepsilon,\delta}_{y,\mathcal{D}_{j^*+1}}$ to τ and let b_{x,j^*+1}, b_{y,j^*+1} denote the measurement outcomes we obtain. Then,

$$\Pr\left[b_{x,j^*} - b_{x,j^*+1} > \frac{29\gamma}{32k} \wedge b_{y,j^*} - b_{y,j^*+1} > \frac{29\gamma}{32k}\right] > \frac{1}{4p(\lambda)\cdot k^3(\lambda)}$$

where the probability is taken over the randomness of the challenger, the adversary $\mathcal{A}_0'$ and the measurement outcomes.

First we define the following notation.

Definition 28. *Let $\mathsf{Exp}_{\mathcal{C},\ell}$ denote the outcome of the following experiment where $\mathcal{C}$ is a ciphertext distribution that can depend on pp.*

1. *Uniformly at random sample x, y, j^* such that $1 \le x < y \le k+1$ and $j^* \in \{1, \dots, k\}$.*

2. *Execute $pk, sk \leftarrow \mathsf{PKE.Setup}(1^\lambda)$.*

3. *Simulate the first step of $\mathcal{A}_0'$ and the challenger of $\mathcal{G}$:*

   3.1. *Simulate $\mathcal{A}$ on pk by sampling a quantum secret key (as in Hyb_1) whenever $\mathcal{A}$ makes a query, and submitting the key to it. Let $\mathsf{R}_{\mathsf{adv}}, (m_\ell^0, m_\ell^1)_{\ell\in[k+1]}, (U_\ell)_{\ell\in[k+1]}$ be the output of $\mathcal{A}$.*

4. *Set* $pp = (x, y, j^*, (id_j)_{j\in[k+1]}, (m^0_\ell, m^1_\ell)_{\ell\in[k+1]}, (U_\ell)_{\ell\in[k+1]}, pk)$.

5. *Sample* $b \leftarrow \{0,1\}$.

6. *Sample* $ct \leftarrow \mathcal{C}(pp, m^b_\ell)$.

7. *Output* $\mathsf{R}_{\mathsf{adv}}, (b, ct), pp$.

We will write $\mathsf{Exp}_{\mathcal{C},\ell} \approx^c_\nu \mathsf{Exp}_{\mathcal{C}',\ell}$ *to denote that the advantage of any computational adversary in distinguishing the outcomes of these experiments is* ν*.*

Proof of Claim 5. Consider instead the following modified version of $\mathcal{A}'_0$. We run $\mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}_i}$ in succession from $i = 0$ to $i = k+1$ on all registers $\ell \in [k+1]$ of $\mathsf{R}_{\mathsf{adv}}$, to obtain values $b'_{\ell,i}$. While the ordering of execution between the registers does not matter, since local operations on disjoint registers commute, for convenience, assume that we run $\mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}_i}$ on all registers before moving onto $\mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}_{i+1}}$. Let ρ_i denote the post-measurement state after having run $\mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}_i}$ on all sub-registers.

First, we claim that

$$\Pr\left[\forall \ell \in [k+1] \;\; b'_{\ell,k+1} < 1/2 + \frac{2\gamma}{32} \middle| \forall \ell \in [k+1] \forall i \in \{0,\dots,k\} \;\; b'_{\ell,i} = b''_{\ell,i}\right] \geq 1-(k(\lambda)+1)\cdot\exp(-\lambda). \tag{1}$$

for any fixed tuple of values $(b''_{\ell,i})_{\ell\in[k+1],i\in\{0,\dots,k\}}$ in the joint support of $(b'_{\ell,i})_{\ell\in[k+1],i\in\{0,\dots,k\}}$. To prove this, we will instead prove the more general statement that for any quantum state ξ of appropriate dimension, we have

$$\Pr\left[\forall \ell \in [k+1] \;\; x_\ell < \frac{1}{2} + \frac{2\gamma}{32}\right] \geq 1 - (k(\lambda)+1)\cdot\exp(-\lambda).$$

where $(x_\ell)_{\ell\in[k+1]} \leftarrow \left(\bigotimes_{\ell\in[k+1]} \mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}_{k+1}}\right)\cdot\xi$.

Let ι be any quantum state of appropriate dimension. By Theorem 17, we have for all $\ell \in [k+1]$

$$\begin{aligned}&\Pr\left[\left(\mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}_{k+1}}\right)\cdot\iota \geq \frac{1}{2} + \frac{2\gamma}{32}\right] \\ \leq &\Pr\left[\left(\mathsf{PI}_{\ell,\mathcal{D}_{k+1}}\right)\cdot\iota \geq \frac{1}{2} + \frac{\gamma}{32}\right] + \exp(-\lambda).\end{aligned}$$

Then, by Theorem 13, we have that if the outcome of $\mathsf{PI}_{\ell,\mathcal{D}_{k+1}}$ is p', then the post-measurement state has success probability p' for the distribution $\mathcal{D}_{k+1}$. However, the challenge ciphertext sampled according to $\mathcal{D}_{k+1}$ is independent of the challenge bit b, hence we always have $p' \leq 1/2$. Hence,

$$\Pr\left[\left(\mathsf{PI}_{\ell,\mathcal{D}_{k+1}}\right)\cdot\iota \geq \frac{1}{2} + \frac{\gamma}{32}\right] = 0.$$

Therefore, $\Pr\left[\left(\mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}_{k+1}}\right)\cdot\iota \geq \frac{1}{2} + \frac{2\gamma}{32}\right] \leq \exp(-\lambda)$. Now, if we apply $\mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}_{k+1}}$ to each part $\xi[i]$, even conditioned on some outcome obtained for the other parts, we get that the result will be $\geq 1/2 + 2\gamma/32$ with probability at most $\exp(-\lambda)$, since we showed the result above for any state ι. Hence, probability of obtaining an outcome $\geq 1/2 + 2\gamma/32$ for at least one part is at most $(k(\lambda)+1)\cdot\exp(-\lambda)$. This gives the desired result (Equation (1)).

Now, we claim that we have $b'_{\ell,1} \geq \frac{1}{2} + \frac{31\gamma}{32}$ for all $\ell \in [k+1]$ with probability $1/(2p(\lambda))$. First, by assumption we have

$$\Pr\left[\left(\bigotimes_{\ell\in[k+1]} \mathsf{ATI}^{\varepsilon,\delta}_{\ell,\mathcal{D},1/2+\frac{31\gamma}{32}}\right)\sigma\right] \geq 1/p(\lambda).$$

since this is exactly the winning condition in Hyb_2. While we later apply other measurements, they do not change the marginal distribution of the initial measurement since we cannot signal backwards in time.

Assume for now that $\mathsf{Exp}_{\mathcal{D},\ell} \approx^c \mathsf{Exp}_{\mathcal{D}_1,\ell}$ for all $\ell \in [k+1]$ and we will prove it later (Claim 13). Then, by Theorem 16 and by above we get

$$\Pr\left[\left(\bigotimes_{\ell\in[k+1]} \mathsf{ATI}^{\varepsilon,\delta}_{\ell,\mathcal{D}_1,1/2+\frac{31\gamma}{32}}\right)\sigma\right] \tag{2}$$

$$\geq \Pr\left[\left(\bigotimes_{\ell\in[k+1]} \mathsf{ATI}^{\varepsilon,\delta}_{\ell,\mathcal{D},1/2+\frac{31\gamma}{32}}\right)\sigma\right] - \mathsf{negl}(\lambda) > 1/(2\cdot p(\lambda)). \tag{3}$$

In Theorem 16, it is easy to see $\mathsf{Exp}_{\mathcal{D},\ell}$ corresponds to $(\mathcal{S},\mathcal{D})$ and $\mathsf{Exp}_{\mathcal{D}_1,\ell}$ corresponds to $(\mathcal{S},\mathcal{D}_1)$; while the measurement results $\vec{p_0}, \vec{p_1}$ correspond to $\bigotimes_{\ell\in[k+1]} \mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}}\sigma$ and $\bigotimes_{\ell\in[k+1]} \mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}_1}\sigma$ when we define our collection of measurements as in Definition 25. That is, our measurement is executing the given state as a decryptor using $\mathsf{U}_{quantum}$ and comparing the outcome to b.

Finally, by combining Equation (1) and Equation (2), we get that with probability at least $1/(4\cdot p(\lambda))$, we have that $\frac{1}{2}+\frac{31\gamma}{32} \leq b'_{\ell,1}$ and $b'_{\ell,k+1} < \frac{1}{2}+\frac{2\gamma}{32}$ for all $\ell \in [k+1]$. Hence, we see that with probability at least $\frac{1}{4p(\lambda)}$; for all $\ell \in [k+1]$ there is $i_\ell \in \{1,\ldots,k\}$ such that $b'_{\ell,i_\ell} - b'_{\ell,i_\ell+1} > \frac{29\gamma}{32k}$. Then, by pigeonhole principle, there is $\ell \neq \ell'$ such that $i_\ell = i_{\ell'}$.

We claim that for any fixed $x < y \in [k+1]$ and $j^* \in [k]$, the marginal distribution (i.e., the reduced density matrix) of $\rho_{j^*}[x,y], (b'_{\ell,0})_{\ell\in[k+1]}, (b'_{x,i})_{i\in[j^*+1]}, (b'_{y,i})_{i\in[j^*+1]}$ in the above experiment is the same as the distribution of $\tau, (b_{\ell,0})_{\ell\in[k+1]}, (b_{x,i})_{i\in[j^*+1]}, (b_{y,i})_{i\in[j^*+1]}$ conditioned on the fixed values of x, y, j^*. This follows from two arguments. First, no-signalling between disjoint registers gives that whether or not we apply measurements on the other registers does not change the marginal distributions of measurement outcomes and post-measurement states on registers x, y. Similarly, by the time we are applying measurements for $\mathcal{D}_i$ for $i \geq j^*+1$, the measurement outcomes for $\mathcal{D}_{j^*}$ are already determined. Since it is not possible to signal backwards in time, the marginal distributions for measurement outcomes b_{ℓ,j^*} is not affected by whether or not we apply the measurements for $\mathcal{D}_i$ for $i \geq j^*+1$.

We have already shown that with probability $1/(4p(\lambda))$, there is guaranteed to be a *jump* in measurement results. Since x, y, j^* are sampled independently by $\mathcal{A}'_0$, they hit the correct indices ℓ, ℓ' satisfying $i_\ell = i_{\ell'}$ with probability $1/k\binom{k+1}{2}$ and j^* hits $i_\ell = i_{\ell'}$ with probability $1/k$. Therefore, we finally have

$$\Pr\left[b_{x,j^*} - b_{x,j^*+1} > \frac{29\gamma}{32k} \wedge b_{y,j^*} - b_{y,j^*+1} > \frac{29\gamma}{32k}\right] > \frac{1}{4\cdot p(\lambda)\cdot k^3(\lambda)}.$$

□

We have shown that the freeloaders x, y use the same identity string block $[id_{j^*}, id_{j^*+1}-1]$ to decrypt. Now we will further show that they use the exact same identity string id_{j^*}. To that end, we first define some intermediary challenge ciphertext distributions. Define the following for all $j \in \{0,1,\ldots,k\}$ and $\Delta \in \{0,1,\ldots,id_{j+1}-id_j-1\}$. For notational convenience, also define $\mathcal{D}_j^{id_{j+1}-id_j,0}$ to be $\mathcal{D}_{j+1}^{(0,0)}$ for all $j \in \{0,1,\ldots,k\}$. Also note that $\mathcal{D}_j^{(0,0)}$ is exactly the same as $\mathcal{D}_j$ for $j \in [k]$.

- $\underline{\mathcal{D}_j^{(\Delta,0)}(m)}$:

1. Sample $r \leftarrow \{0,1\}^{c_L(\lambda)}$.
2. Sample a PRF key K_2 for $F_2.\mathsf{KeyGen}(1^\lambda)$.
3. Sample $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}^{(j,\Delta,0)}_{\mathsf{OPMem},cpk,K_2,r,m,id_j+\Delta})$.

```
$\mathsf{PCt}^{(j,\Delta,0)}_{\mathsf{OPMem},cpk,K_2,r,m,id_j+\Delta}(id, u_1, \ldots, u_{c_L(\lambda)})$
Hardcoded: $\mathsf{OPMem}, cpk, K_2, r, m, id_j + \Delta$
(a) Run $\mathsf{OPMem}(id, u_1, \ldots, u_{c_L(\lambda)}, r)$. If it outputs 0, output $\perp$ and terminate.
(b) If $id < id_j + \Delta$, set $a = \top$. Otherwise, set $a = m$.
(c) Output $\mathsf{IBE.Enc}(cpk, id, a; F_2(K_2, id))$.
```

4. Output (OPCt, r).

- $\mathcal{D}_j^{(\Delta,1)}(m)$:

1. Sample $r \leftarrow \{0,1\}^{c_L(\lambda)}$.
2. Sample a PRF key K_2 for $F_2.\mathsf{KeyGen}(1^\lambda)$.
3. $ct^* = \mathsf{IBE.Enc}(cpk, id_j + \Delta, m; F_2(K_2, id_j + \Delta))$.
4. $K_2\{id_j + \Delta\} \leftarrow F_2.\mathsf{Punc}(K_2, id_j + \Delta)$.
5. Sample $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}^{(j,\Delta,1)}_{\mathsf{OPMem},cpk,K_2\{id_j+\Delta\},r,m,id_j+\Delta,ct^*})$.

```
$\mathsf{PCt}^{(j,\Delta,1)}_{\mathsf{OPMem},cpk,K_2\{id_j+\Delta\},r,m,id_j+\Delta,ct^*}(id, u_1, \ldots, u_{c_L(\lambda)})$
Hardcoded: $\mathsf{OPMem}, cpk, K_2\{id_j + \Delta\}, r, m, id_j + \Delta, ct^*$
(a) Run $\mathsf{OPMem}(id, u_1, \ldots, u_{c_L(\lambda)}, r)$. If it outputs 0, output $\perp$ and terminate.
(b) If $id = id_j + \Delta$, output $ct^*$ and terminate.
(c) If $id < id_j + \Delta + 1$, set $a = \top$. Otherwise, set $a = m$.
(d) Output $\mathsf{IBE.Enc}(cpk, id, a; F_2(K_2, id))$.
```

6. Output (OPCt, r).

- $\mathcal{D}_j^{(\Delta,2)}(m)$:

1. Sample $r \leftarrow \{0,1\}^{c_L(\lambda)}$.
2. Sample a PRF key K_2 for $F_2.\mathsf{KeyGen}(1^\lambda)$.
3. Sample z^* uniformly at random from the output space of F_2.
4. $ct^* = \mathsf{IBE.Enc}(cpk, id_j + \Delta, m; z^*)$.
5. $K_2\{id_j + \Delta\} \leftarrow F_2.\mathsf{Punc}(K_2, id_j + \Delta)$.
6. Sample $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}^{(j,\Delta,2)}_{\mathsf{OPMem},cpk,K_2\{id_j+\Delta\},r,m,id_j+\Delta,ct^*})$.

```
$\mathsf{PCt}^{(j,\Delta,2)}_{\mathsf{OPMem},cpk,K_2\{id_j+\Delta\},r,m,id_j+\Delta,ct^*}(id, u_1, \ldots, u_{c_L(\lambda)})$
Hardcoded: $\mathsf{OPMem}, cpk, K_2\{id_j + \Delta\}, r, m, id_j + \Delta, ct^*$
(a) Run $\mathsf{OPMem}(id, u_1, \ldots, u_{c_L(\lambda)}, r)$. If it outputs 0, output $\perp$ and terminate.
```

```
(b) If $id = id_j + \Delta$, output $ct^*$ and terminate.
(c) If $id < id_j + \Delta + 1$, set $a = \top$. Otherwise, set $a = m$.
(d) Output $\mathsf{IBE.Enc}(cpk, id, a; F_2(K_2, id))$.
```

7. Output (OPCt, r).

- $\underline{\mathcal{D}_j^{(\Delta,3)}(m)}$:

1. Sample $r \leftarrow \{0,1\}^{c_L(\lambda)}$.
2. Sample a PRF key K_2 for $F_2.\mathsf{KeyGen}(1^\lambda)$.
3. Sample z^* uniformly at random from the output space of F_2.
4. $ct^* = \mathsf{IBE.Enc}(cpk, id_j + \Delta, m; z^*)$.
5. $K_2\{id_j + \Delta\} \leftarrow F_2.\mathsf{Punc}(K_2, id_j + \Delta)$.
6. Compute $(A_i^*, s_i^*, s_i^{'*}) = F_1(K_1, id_j + \Delta)$.
7. For $i \in [c_L(\lambda)]$, set $g_i = \mathsf{Can}_{A_i^*}$ if $(r)_i = 0$ and set $g_i = \mathsf{Can}_{(A_i^*)^\perp}$ if $(r)_i = 1$.
8. For $i \in [c_L(\lambda)]$, compute $y_i = g_i(s_i^*)$ if $(r)_i = 0$ and $y_i = g_i(s_i^{'*})$ if $(r)_i = 1$.
9. Set g to be the function $g(v_1, \ldots, v_{c_L(\lambda)}) = (g_1(v_1) || \ldots || g_{c_L(\lambda)}(v_{c_L(\lambda)}))$.
10. Set $y = y_1 || \ldots || y_{c_L(\lambda)}$.
11. $\mathsf{OCC} \leftarrow \mathsf{CCObf.Obf}(g, y, ct^*)$.
12. Sample $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}^{(j,\Delta,3)}_{\mathsf{OPMem},cpk,K_2\{id_j\},r,m,id_j+\Delta,\mathsf{OCC}})$.

```
$\mathsf{PCt}^{(j,\Delta,3)}_{\mathsf{OPMem},cpk,K_2\{id_j+\Delta\},r,m,id_j+\Delta,\mathsf{OCC}}(id, u_1, \ldots, u_{c_L(\lambda)})$
Hardcoded: $\mathsf{OPMem}, cpk, K_2\{id_j + \Delta\}, r, m, id_j + \Delta, \mathsf{OCC}$
(a) If $id = id_j + \Delta$, output the output of $\mathsf{OCC}(u_1, \ldots, u_{c_L(\lambda)})$ and terminate.
(b) Run $\mathsf{OPMem}(id, u_1, \ldots, u_{c_L(\lambda)}, r)$. If it outputs 0, output $\perp$ and terminate.
(c) If $id < id_j + \Delta + 1$, set $a = \top$. Otherwise, set $a = m$.
(d) Output $\mathsf{IBE.Enc}(cpk, id, a; F_2(K_2, id))$.
```

13. Output (OPCt, r).

- $\underline{\mathcal{D}_j^{(\Delta,4)}(m)}$: Same as $\mathcal{D}_j^{(\Delta,3)}$ except for the following. Replace the line

$$ct^* = \mathsf{IBE.Enc}(cpk, id_j + \Delta, m; z^*)$$

with

$$ct^* = \mathsf{IBE.Enc}(cpk, id_j + \Delta, \top; z^*).$$

- $\underline{\mathcal{D}_j^{(\Delta,5)}(m)}$: Same as $\mathcal{D}_j^{(\Delta,2)}$ except for the following. Replace the line

$$ct^* = \mathsf{IBE.Enc}(cpk, id_j + \Delta, m; z^*)$$

with

$$ct^* = \mathsf{IBE.Enc}(cpk, id_j + \Delta, \top; z^*).$$

- $\underline{\mathcal{D}_j^{(\Delta,6)}(m)}$: Same as $\mathcal{D}_j^{(\Delta,1)}$ except for the following. Replace the line

  $$ct^* = \mathsf{IBE.Enc}(cpk, id_j + \Delta, m; F_2(K_2, id_j + \Delta))$$

  with

  $$ct^* = \mathsf{IBE.Enc}(cpk, id_j + \Delta, \top; F_2(K_2, id_j + \Delta)).$$

Now, we show that these distributions *collapse* around $\Delta = 0$ for each j. Below, all our indistinguishability claims are for $2^{5\lambda} \cdot 2^{8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$-time adversaries and we set $\nu(\lambda) = 2^{-6\lambda} \cdot 2^{-8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$.

Claim 6. $\mathsf{Exp}_{\mathcal{D}_j^{(\Delta,0)},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(\Delta,1)},\ell}$ *for all* $j \in \{0, 1, \ldots, k\}$*,* $\Delta \in \{0, 1, \ldots, id_{j+1} - id_j - 1\}$ *and* $\ell \in [k + 1]$*.*

Proof. Observe that by punctured key correctness of F_2 (Definition 1), the different obfuscated programs $\mathsf{PCt}^{(j,\Delta,0)}_{\mathsf{OPMem},cpk,K_2,r,m,id_j+\Delta}$ and $\mathsf{PCt}^{(j,\Delta,1)}_{\mathsf{OPMem},cpk,K_2\{id_j+\Delta\},r,m,id_j+\Delta,ct^*}$ in these hybrids have the same functionality. The result follows by security of $i\mathcal{O}$ and by our choice of parameters. □

Claim 7. $\mathsf{Exp}_{\mathcal{D}_j^{(\Delta,1)},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(\Delta,2)},\ell}$ *for all* $j \in \{0, 1, \ldots, k\}$*,* $\Delta \in \{0, 1, \ldots, id_{j+1} - id_j - 1\}$ *and* $\ell \in [k + 1]$*.*

Proof. The result follows by selective puncturing security of F_2 (Definition 1) and our choice of parameters. □

Claim 8. $\mathsf{Exp}_{\mathcal{D}_j^{(\Delta,2)},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(\Delta,3)},\ell}$ *for all* $j \in \{0, 1, \ldots, k\}$*,* $\Delta \in \{0, 1, \ldots, id_{j+1} - id_j - 1\}$ *and* $\ell \in [k + 1]$*.*

Proof. Observe that the obfuscated ciphertext programs PCt in these hybrids have the same functionality by correctness of CCObf, since a vector w is in $A_i^* + s_i^*$ if and only if $\mathsf{Can}_{A_i^*}(w) = \mathsf{Can}_{A_i^*}(s_i^*)$ and similarly for $(A^*)_i^\perp + s_i'^*$. Then, the claim follows by the security of $i\mathcal{O}$. □

Claim 9. $\mathsf{Exp}_{\mathcal{D}_j^{(\Delta,3)},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(\Delta,4)},\ell}$ *if*

- $j \in \{1, \ldots, k\}$ *and* $\Delta \in \{1, \ldots, id_{j+1} - id_j - 1\}$*, or*
- $j = 0$ *and* $\Delta \in \{0, 1, \ldots, id_{j+1} - id_j - 1\}$

and for all $\ell \in [k + 1]$*.*

Proof. Observe that in these hybrids, the randomness used to invoke $\mathsf{IBE.Enc}$ to compute ct^* is uniformly and independently sampled. Further, the adversary only has the IBE keys for the identities $id_1, id_2, \ldots, id_k$, all of which are different from the identity $id_j + \Delta$ under which ct^* is encrypted. Hence, by IBE security (Definition 15), the result follows. □

Claim 10. $\mathsf{Exp}_{\mathcal{D}_j^{(\Delta,4)},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(\Delta,5)},\ell}$ *for all* $j \in \{0, 1, \ldots, k\}$*,* $\Delta \in \{0, 1, \ldots, id_{j+1} - id_j - 1\}$ *and* $\ell \in [k + 1]$*.*

Proof. Essentially the same argument as in Claim 8 yields the result. □

Claim 11. $\mathsf{Exp}_{\mathcal{D}_j^{(\Delta,5)},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(\Delta,6)},\ell}$ *for all* $j \in \{0, 1, \ldots, k\}$*,* $\Delta \in \{0, 1, \ldots, id_{j+1} - id_j - 1\}$ *and* $\ell \in [k + 1]$*.*

Proof. Essentially the same argument as in Claim 7 yields the result. □

Claim 12. *$\mathsf{Exp}_{\mathcal{D}_j^{(\Delta,6)},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(\Delta+1,0)},\ell}$ for all $j \in \{0,1,\dots,k\}$, $\Delta \in \{0,1,\dots,id_{j+1} - id_j - 1\}$ and $\ell \in [k+1]$.*

Proof. Essentially the same argument as in Claim 6 yields the result. □

Claim 13. *For all $\ell \in [k+1]$, we have*

- $\mathsf{Exp}_{\mathcal{D}_0,\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_1,\ell}$
- $\mathsf{Exp}_{\mathcal{D}_j^{(0,4)},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_{j+1},\ell}$ *for all $j \in \{0,1,\dots,k\}$*
- $\mathsf{Exp}_{\mathcal{D}_j,\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(0,3)},\ell}$ *for all $j \in \{0,1,\dots,k\}$*

where $\nu(\lambda) = 2^{-5\lambda} \cdot 2^{-8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$.

Proof. It is easy to see that $\mathcal{D}_0 \approx^c_{\nu(\lambda)} \mathcal{D}_0^{(0,0)}$ and $\mathcal{D}_{k+1} \approx^c_{\nu(\lambda)} \mathcal{D}_{k+1}^{(0,0)}$ by the security of $i\mathcal{O}$.

Rest follows by a simple calculation using the above results. □

Definition 29. *We will write $\mathcal{D}'$ to denote $\mathcal{D}_{j^*}^{(0,3)}$ and $\mathcal{D}''$ to denote $\mathcal{D}_{j^*}^{(0,4)}$ where j^* is as output by $\mathcal{A}'_0$.*

Finally, we show that the success probabilities for both freeloaders jump exactly at j^*.

Claim 14. *Let τ be the bipartite state output by $\mathcal{A}'_0$ in $\mathcal{G}$. Let p'_x, p'_y be the outcome of applying $\mathsf{PI}_{x,\mathcal{D}'} \otimes \mathsf{PI}_{y,\mathcal{D}'}$ to τ. Similarly, let p''_x, p''_y be the outcome of applying $\mathsf{PI}_{x,\mathcal{D}''} \otimes \mathsf{PI}_{y,\mathcal{D}''}$ to τ. Then,*

- $\Pr\left[p'_x > b_{x,j^*} - \frac{3\gamma}{32k} \wedge p'_y > b_{y,j^*} - \frac{3\gamma}{32k}\right] \geq 1 - 2^{-2\lambda} \cdot 2^{-4\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$.
- $\Pr\left[b_{x,j^*} - p''_x > \frac{28\gamma}{32k} \wedge b_{y,j^*} - p''_y > \frac{28\gamma}{32k}\right] > \frac{1}{q(\lambda)}$ *for some polynomial $q(\cdot)$.*

Proof. Let (a'_x, a'_y) be the outcome of applying $\mathsf{API}^{\varepsilon,\delta}_{x,\mathcal{D}_{j^*}} \otimes \mathsf{API}^{\varepsilon,\delta}_{y,\mathcal{D}_{j^*}}$ to τ. Then, by Theorem 19, Theorem 18 and by definition of b_{x,j^*}, b_{y,j^*}, we have

$$\Pr\left[a'_x > b_{x,j^*} - \frac{3\gamma}{32k} \wedge a'_y > b_{y,j^*} - \frac{3\gamma}{32k}\right] \geq 1 - \mathsf{poly}(\lambda) \cdot \delta(\lambda).$$

Then, since $\mathsf{Exp}_{\mathcal{D}_{j^*},\ell} \approx^c_{\nu} \mathsf{Exp}_{\mathcal{D}',\ell}$ against $2^{5\lambda} \cdot 2^{8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$-time adversaries where $\nu(\lambda) = 2^{-5\lambda} \cdot 2^{-8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$, we get

$$\Pr\left[p'_x > b_{x,j^*} - \frac{3\gamma}{32k} \wedge p'_y > b_{y,j^*} - \frac{3\gamma}{32k}\right] \geq 1 - 2^{-2\lambda} \cdot 2^{-4\lambda^{0.3C_{\mathsf{MoE.Coll}}}}.$$

by Theorem 16.

See the proof of Claim 5 for a remark on how to invoke Theorem 16. Note that here, we are applying the measurements to τ rather than to the state σ. However, since the procedure that gives τ from σ is an efficient procedure that only uses *pp*, the indistinguishability between $\mathcal{D}_{j^*}$ and $\mathcal{D}'$ given σ still applies when we are instead given τ, hence Theorem 16 indeed applies.

We now move onto the second claim. By Claim 5, we have that

$$\Pr\left[b_{x,j^*} - b_{x,j^*+1} > \frac{29\gamma}{32k} \wedge b_{y,j^*} - b_{y,j^*+1} > \frac{29\gamma}{32k}\right]$$

is non-negligible. Further, we have $\mathsf{Exp}_{\mathcal{D}_{j^*+1},\ell} \approx \mathsf{Exp}_{\mathcal{D}'',\ell}$. By Theorem 16 and Theorem 20, we get that

$$\Pr\left[b_{x,j^*} - p''_x > \frac{28\gamma}{32k} \wedge b_{y,j^*} - p''_y > \frac{28\gamma}{32k}\right]$$

is non-negligible. Similar to above, the indistinguishability of $\mathcal{D}_{j^*+1}$ and $\mathcal{D}''$ given σ still applies when we are given the state τ instead. Therefore, Theorem 16 indeed applies. □

Extracting MoE Vectors

We have shown that the two freeloaders use the same identity j^* to decrypt. Now we will show that we can exract MoE vectors from these two freeloaders *simultaneously.* That is, we extract MoE vectors from one of the freeloaders even conditioned on successful extraction from the other one.[34]

Claim 15. *There exist efficient $\mathcal{A}'_1, \mathcal{A}'_2$ such that $(\mathcal{A}'_0, \mathcal{A}'_1, \mathcal{A}'_2)$ wins $\mathcal{G}$ with probability $\frac{1}{2^{0.4\cdot\lambda} C_{\mathsf{MoE.Coll}}}$.*

Proof. For a challenge ciphertext distribution $\mathcal{C}$, let $\mathsf{Exp}'_{\mathcal{C},x}$ denote the outcome of the following experiment.

1. Execute $pk, sk \leftarrow \mathsf{PKE.Setup}(1^\lambda)$.
2. Simulate $\mathcal{A}'_0$ and the challenger of $\mathcal{G}$:
   - 2.1. Simulate $\mathcal{A}$ on pk by sampling a quantum secret key (as in Hyb_1) whenever $\mathcal{A}$ makes a query, and submitting the key to it. Let $\mathsf{R}_{\mathsf{adv}}, (m^0_\ell, m^1_\ell)_{\ell\in[k+1]}, (U_\ell)_{\ell\in[k+1]}$ be the output of $\mathcal{A}$.
   - 2.2. Uniformly at random sample x, y, j^* such that $1 \le x < y \le k+1$ and $j^* \in \{1, \ldots, k\}$.
   - 2.3. Apply $\mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}_0}$ to all registers $\mathsf{R}_{\mathsf{adv}}[\ell]$ for $\ell \in [k+1]$, let $b_{\ell,0}$ be the measurement outcomes.
   - 2.4. Apply $\mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}_i}$ in succession for $i = 1$ to j^* to $\mathsf{R}_{\mathsf{adv}}[x]$, let $b_{x,i}$ be the measurement outcomes.
   - 2.5. Apply $\mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}_i}$ in succession for $i = 1$ to j^* to $\mathsf{R}_{\mathsf{adv}}[y]$, let $b_{y,i}$ be the measurement outcomes.
3. Set $pp = (x, y, j^*, (id_j)_{j\in[k+1]}, (m^0_\ell, m^1_\ell)_{\ell\in[k+1]}, (U_\ell)_{\ell\in[k+1]}, pk)$.
4. Sample $b \leftarrow \{0, 1\}$.
5. Sample $ct \leftarrow \mathcal{C}(pp, m^b_\ell)$.
6. Output $\mathsf{R}_{\mathsf{adv}}[x], (b, ct), pp$.

It is easy to see that $\mathsf{Exp}_{\mathcal{C}_0,\ell} \approx^c_\nu \mathsf{Exp}_{\mathcal{C}_1,\ell}$ implies $\mathsf{Exp}'_{\mathcal{C}_0,x} \approx^c_\nu \mathsf{Exp}'_{\mathcal{C}_1,x}$ since they only differ in their auxiliary states and we can efficiently obtain the auxiliary state of the latter using the auxiliary state of the former.

By Claim 14, we have

[34] Note that this is not a direct consequence of extraction from a single freeloader due to entanglement.

1. $\Pr\left[\mathsf{PI}_{x,\mathcal{D}'} \cdot \tau[1] \leq b_{x,j^*} - \frac{3\gamma}{32k}\right] \leq 2^{-2\lambda} \cdot 2^{-4\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$, and

2. $\Pr\left[\mathsf{PI}_{x,\mathcal{D}''} \cdot \tau[1] < b_{x,j^*} - \frac{28\gamma}{32k}\right]$ is non-negligible.

Suppose for a contradiction that $\mathsf{Exp}'_{\mathcal{D}',x} \approx^c \mathsf{Exp}'_{\mathcal{D}'',x}$. Then, by Theorem 17 and Theorem 16, Item 2 implies that

$$\Pr\left[\mathsf{PI}_{x,\mathcal{D}'} \cdot \tau[1] < b_{x,j^*} - \frac{26\gamma}{32k}\right]$$

is non-negligible. This is a contradiction to Item 1, therefore, $\mathsf{Exp}'_{\mathcal{D}',x} \not\approx^c \mathsf{Exp}'_{\mathcal{D}'',x}$. We define the distribution $\mathcal{D}_{sim}$ by modifying $\mathcal{D}'$ as follows: We replace the line

$$\mathsf{OCC} \leftarrow \mathsf{CCObf.Obf}(g, y, ct^*)$$

with

$$\mathsf{OCC} \leftarrow \mathsf{CCObf.Sim}(1^\lambda, |g|, |y|, |ct^*|).$$

Since $\mathsf{Exp}'_{\mathcal{D}',x} \not\approx^c \mathsf{Exp}'_{\mathcal{D}'',x}$, we have either $\mathsf{Exp}'_{\mathcal{D}',x} \not\approx^c \mathsf{Exp}'_{\mathcal{D}_{sim},x}$ or $\mathsf{Exp}'_{\mathcal{D}'',x} \not\approx^c \mathsf{Exp}'_{\mathcal{D}_{sim},x}$. We will only discuss the first case but the second case follows from the same argument.

Now, we will give a distribution $\mathcal{B}$ over compute-and-compare programs (with quantum auxiliary information) and an adversary $\mathcal{A}_{CC}$ that breaks the security of CCObf for this distribution. This in turn will mean by Definition 9 that there is an adversary that can predict the *target* value of these programs, given the description of the compute part of the program and the auxiliary information.

We first define the distribution $\mathcal{B}$.

$\underline{\mathcal{B}(1^\lambda)}$

1. Execute $pk, sk \leftarrow \mathsf{PKE.Setup}(1^\lambda)$.

2. Simulate $\mathcal{A}'_0$ and the challenger of $\mathcal{G}$:

   2.1. Simulate $\mathcal{A}$ on pk by sampling a quantum secret key (as in Hyb_1) whenever $\mathcal{A}$ makes a query, and submitting the key to it. Let $\mathsf{R_{adv}}, (m^0_\ell, m^1_\ell)_{\ell\in[k+1]}, (U_\ell)_{\ell\in[k+1]}$ be the output of $\mathcal{A}$.

   2.2. Uniformly at random sample x, y, j^* such that $1 \leq x < y \leq k+1$ and $j^* \in \{1, \dots, k\}$.

   2.3. Apply $\mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}_0}$ to all registers $\mathsf{R_{adv}}[\ell]$ for $\ell \in [k+1]$, let $b_{\ell,0}$ be the measurement outcomes.

   2.4. Apply $\mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}_i}$ in succession for $i = 1$ to j^* to $\mathsf{R_{adv}}[x]$, let $b_{x,i}$ be the measurement outcomes.

   2.5. Apply $\mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}_i}$ in succession for $i = 1$ to j^* to $\mathsf{R_{adv}}[y]$, let $b_{y,i}$ be the measurement outcomes.

3. Set $pp = (x, y, j^*, (id_j)_{j\in[k+1]}, (m^0_\ell, m^1_\ell)_{\ell\in[k+1]}, (U_\ell)_{\ell\in[k+1]}, pk)$.

4. Sample $b \leftarrow \{0, 1\}$.

5. Simulate the first steps of $\mathcal{D}'$ on m^b_x:

   5.1. Sample $r \leftarrow \{0, 1\}^{c_L(\lambda)}$.

   5.2. Sample z^* uniformly at random the output space of F_2.

   5.3. $ct^* = \mathsf{IBE.Enc}(cpk, id_j, m^b_x; z^*)$.

   5.4. Compute $(A_i^*, s_i^*, s_i^{'*}) = F_1(K_1, id_{j^*})$.

   5.5. For $i \in [c_L(\lambda)]$, set $g_i = \mathsf{Can}_{A_i^*}$ if $(r)_i = 0$ and set $g_i = \mathsf{Can}_{(A_i^*)^\perp}$ if $(r)_i = 1$.

   5.6. For $i \in [c_L(\lambda)]$, compute $y_i = g_i(s_i^*)$ if $(r)_i = 0$ and $y_i = g_i(s_i^{'*})$ if $(r)_i = 1$.

   5.7. Set g to be the function $g(v_1, \dots, v_{c_L(\lambda)}) = (g_1(v_1)|| \dots ||g_{c_L(\lambda)}(v_{c_L(\lambda)}))$.

   5.8. Set $y = y_1|| \dots ||y_{c_L(\lambda)}$.

6. Output (g, y, ct^*) as the compute-and-compare program and

$$(\mathsf{R}_{\mathsf{adv}}[x], pp, r, m_x^b, id_{j^*}, b)$$

   as the auxiliary information.

We define the adversary $\mathcal{A}_{CC}$ as follows. Let $\mathcal{A}_{dist}$ be an adversary that distinguishes $\mathsf{Exp}_{\mathcal{D}',x} \not\approx^c \mathsf{Exp}_{\mathcal{D}_{sim},x}$.

$\underline{\mathcal{A}_{CC}(P, \mathsf{R}_{\mathsf{aux}})}$

1. Parse $(\mathsf{R}, pp, r, m_x^b, id_{j^*}, b) = \mathsf{R}_{\mathsf{aux}}$.
2. Parse $(x, y, j^*, (id_j)_{j\in[k+1]}, (m_\ell^0, m_\ell^1)_{\ell\in[k+1]}, (U_\ell)_{\ell\in[k+1]}, pk) = pp$.
3. Sample a PRF key K_2 for $F_2.\mathsf{KeyGen}(1^\lambda)$.
4. Sample $K_2\{id_{j^*}\} \leftarrow F_2.\mathsf{Punc}(K_2, id_{j^*})$.
5. Sample $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt})$.

```
PCt(id, u_1, ..., u_{c_L(λ)})
Hardcoded: OPMem, cpk, K_2{id_{j*}}, r, m_x^b, id_{j*}, P
  (a) If $id = id_{j^*}$, output the output of $P(u_1, \dots, u_{c_L(\lambda)})$ and terminate.
  (b) Run $\mathsf{OPMem}(id, u_1, \dots, u_{c_L(\lambda)}, r)$. If it outputs 0, output $\perp$ and terminate.
  (c) If $id < id_{j^*} + 1$, set $a = \top$. Otherwise, set $a = m_x^b$.
  (d) Output $\mathsf{IBE.Enc}(cpk, id, a; F_2(K_2, id))$.
```

6. Set $ct = (\mathsf{OPCt}, r)$.
7. Output $\mathcal{A}_{dist}(\mathsf{R}, (ct, b), pp)$.

It is easy to see that $\mathcal{A}_{CC}(\mathsf{CCObf.Obf}(g, y, ct^*), \mathsf{R}_{\mathsf{aux}})$ corresponds to $\mathcal{A}_{dist}(\mathsf{Exp}_{\mathcal{D}',x})$ while $\mathcal{A}_{CC}($ $\mathsf{CCObf.Sim}(1^\lambda, |g|, |y|, |ct^*|), \mathsf{R}_{\mathsf{aux}}))$ corresponds to $\mathcal{A}_{dist}(\mathsf{Exp}_{\mathcal{D}_{sim},x})$ where $(g, y, ct^*) \leftarrow \mathcal{B}(1^\lambda)$. Hence, since $\mathcal{A}_{dist}$ distinguishes $\mathsf{Exp}_{\mathcal{D}',x} \not\approx^c \mathsf{Exp}_{\mathcal{D}_{sim},x}$, by Theorem 12 there exists an adversary $\mathcal{M}_1$ that can extract vectors u_i such that $g((u_i)_{i\in c_L(\lambda)}) = y$, using the quantum auxiliary information defined above and the description of g. Note that g can be computed efficiently given r and $(A_i^*)_{i\in[c_L(\lambda)]}$, which are indeed provided to $\mathcal{A}_1'$ in $\mathcal{G}$. Similarly, $\mathcal{A}_1'$ can compute $\mathsf{R}_{\mathsf{aux}}$ from its input provided by $\mathcal{A}_0'$. Therefore, we set $\mathcal{A}_1'$ to be the adversary that computes the input as above and simulates $\mathcal{M}_1$. It is easy to see that $\mathcal{A}_1'$ outputs correct vectors in the game $\mathcal{G}$ with probability at least $2^{-\lambda^{0.2\cdot C_{\mathsf{MoE.Coll}}}}$, since CCObf is a compute-and-compare obfuscation scheme for $2^{-\lambda^{0.2 \cdot C_{\mathsf{MoE.Coll}}}}$-unpredictable distributions.

Now, we will argue that we can simultaneously extract MoE vectors from the second register, that is, we can extract even conditioned on a successful extraction from the first register. Let ξ denote the post-measurement state of the input state of $\mathcal{A}'_2$, conditioned on $\mathcal{A}'_1$ succeeding. First, define $\mathsf{Exp}''_{\mathcal{C},y}$ as follows.

1. Simulate $\mathcal{B}(1^\lambda)$.
2. Run $\mathcal{A}'_1$ on $(\mathsf{R}_{\mathsf{adv}}[x], pp, r, m^b_x, id_{j^*}, b)$ and g to obtain vectors $(u_i)_{i \in c_L(\lambda)}$.
3. Check if $\mathsf{OPMem}(id_{j^*}, u_1, \ldots, u_{c_L(\lambda)}, r)$. If the output is 0, output $\perp$ and terminate.
4. Sample $b \leftarrow \{0, 1\}$.
5. Sample $ct \leftarrow \mathcal{C}(pp, m^b_\ell)$.
6. Output $\mathsf{R}_{\mathsf{adv}}[y], (b, ct), pp$.

Observe that the state of the register $\mathsf{R}_{\mathsf{adv}}[y]$ output above is ξ (when the experiment outcome is not $\perp$). We claim that ξ satisfies

1. $\Pr\left[\mathsf{PI}_{y,\mathcal{D}'} \cdot \xi \leq b_{y,j^*} - \frac{3\gamma}{32k}\right] \leq \frac{3}{2} \cdot \sqrt{2^{-2\lambda} \cdot 2^{-4\lambda^{0.3 C_{\mathsf{MoE.Coll}}}}} \cdot 2^{\lambda^{0.2 \cdot C_{\mathsf{MoE.Coll}}}}$.
2. $\Pr\left[\mathsf{PI}_{y,\mathcal{D}''} \cdot \xi < b_{y,j^*} - \frac{28\gamma}{32k}\right] \geq 2^{-\lambda^{0.3 \cdot C_{\mathsf{MoE.Coll}}}}$.

This first claim follows from Claim 14, Theorem 9, and the fact that extraction on the first register succeeds with probability $\geq 2^{-\lambda^{0.2 \cdot C_{\mathsf{MoE.Coll}}}}$. We argue the second claim as follows. Let E denote the event of successful extraction on the first register, and let G denote the event that applying $\mathsf{PI}_{y,\mathcal{D}''}$ on the second register yields a value $< b_{y,j^*} - \frac{28\gamma}{32k}$. The probability above corresponds to $\Pr[G|E]$, which equals $\frac{\Pr[E|G] \cdot \Pr[G]}{\Pr[E]} \geq \Pr[E|G] \cdot \Pr[G] \geq \Pr[E|G] \cdot \frac{1}{\mathsf{poly}(\lambda)}$. However, observe that we can first apply the measurement $\mathsf{PI}_{y,\mathcal{D}''}$ on the second register, and then try to extract on the first register. Observe that a *gap* still exists on the first register after this measurement on the second register and conditioning on the outcome G, by Claim 14 and Theorem 9, since $\Pr[G] > 1/\mathsf{poly}(\lambda)$. Hence, similar to the extraction argument above, we get that $\Pr[E|G] > 2^{-\lambda^{0.2 \cdot C_{\mathsf{MoE.Coll}}}}$, which proves our claim.

Now, suppose for a contradiction that $\mathsf{Exp}''_{\mathcal{D}',y} \approx^c_\nu \mathsf{Exp}''_{\mathcal{D}'',y}$ against $2^{3\lambda} \cdot 2^{2\lambda^{0.3 C_{\mathsf{MoE.Coll}}}}$-time adversaries where $\nu = 2^{-2\lambda - 1} \cdot 2^{-2\lambda^{0.3 C_{\mathsf{MoE.Coll}}}}$. Then, by Theorem 16, we get that Item 1 implies

$$\Pr\left[\mathsf{PI}_{y,\mathcal{D}''} \cdot \xi \leq b_{y,j^*} - \frac{3\gamma}{32k}\right] \leq 2 \cdot 2^{-\lambda} \cdot 2^{-\lambda^{0.3 \cdot C_{\mathsf{MoE.Coll}}}}.$$

which is a contradiction to Item 2. Hence, $\mathsf{Exp}''_{\mathcal{D}',y} \not\approx^c_\nu \mathsf{Exp}''_{\mathcal{D}'',y}$. Then, using the same extraction argument we used for the first register, by the security of CCObf, we get that there exists an adversary $\mathcal{A}'_2$ such that it outputs the correct coset vectors with probability at least $2^{-\lambda^{0.2 \cdot C_{\mathsf{MoE.Coll}}}}$ *conditioned on* $\mathcal{A}'_1$ *outputting correct coset vectors.* This shows that $(\mathcal{A}'_0, \mathcal{A}'_1, \mathcal{A}'_2)$ wins $\mathcal{G}$ with probability $2^{-0.4 \cdot \lambda^{C_{\mathsf{MoE.Coll}}}}$. □

We have shown that there is an adversary $(\mathcal{A}'_0, \mathcal{A}'_1, \mathcal{A}'_2)$ that wins the game $\mathcal{G}$ with probability

$$1/2^{0.4 \cdot \lambda^{C_{\mathsf{MoE.Coll}}}}.$$

Finally, we show that we can construct an adversary $(\mathcal{A}''_0, \mathcal{A}''_1, \mathcal{A}''_2)$ that can win $\mathsf{MoE} - \mathsf{Coll} - \mathsf{Sel}$.

Claim 16. *There exists efficient* $\mathcal{A}'' = (\mathcal{A}''_0, \mathcal{A}''_1, \mathcal{A}''_2)$ *such that*

$$\Pr\big[\mathsf{MoE} - \mathsf{Coll} - \mathsf{Sel}(\lambda, L(\lambda), \mathcal{A}'') = 1\big] \geq 2^{-0.4 \cdot \lambda^{C_{\mathsf{MoE.Coll}}}}.$$

Proof. $\mathcal{A}''_0$ simulates both the challenger of $\mathcal{G}$ and the adversary $\mathcal{A}'_0$ as follows. It first samples the random collision-free identity strings $id_1 < \cdots < id_k$, and the random index j^* (which it outputs to its challenger); and also $cpk, csmk \leftarrow \mathsf{IBE.Setup}(1^\lambda)$. Then, it sets $pk = (cpk, \mathsf{OPMem})$ where it obtains OPMem from its challenger. Then, whenever $\mathcal{A}'_0$ makes a key query for $i \in [k]$, it queries its own challenger for the coset state associated with $id_{\alpha(i)}$. It also samples ck as in $\mathsf{PKE.QKeyGen}$ using $cmsk$, and submits the coset state, $id_{\alpha(i)}$ and ck to $\mathcal{A}'_0$. Finally, when $\mathcal{A}'_0$ yields a bipartite register, $\mathcal{A}''_0$ outputs it.

We define $\mathcal{A}''_1$ so that it simulates $\mathcal{A}'_1$ and make no queries during the second query phase. $\mathcal{A}''_2$ is defined similarly for $\mathcal{A}'_2$. It is easy to see that $\mathcal{A}''$ playing $\mathsf{MoE} - \mathsf{Coll} - \mathsf{Sel}$ perfectly simulates $\mathcal{G}$ as played by $\mathcal{A}'$, hence $\mathcal{A}''$ wins with probability $2^{-0.4 \cdot \lambda^{C_{\mathsf{MoE.Coll}}}}$. □

This completes the security proof, since the above is a contradiction to Theorem 24.

8 Public-Key Functional Encryption with Copy-Protected Functional Keys

In this section, we formally define functional encryption with copy-protected functional keys. Then, we give a construction based on coset states and prove it secure.

We note that Kitagawa and Nishimaki [KN22] define a simpler model of functional encryption with copy-protected functional keys and give a secure construction with respect to their model. In their model, the adversary can query for any number of functional keys, but only one can be in copy-protected mode. In turn, the adversary only outputs two freeloaders (i.e., only 1 → 2 copy-protection is considered). Further, the freeloader adversaries are not allowed to query for more keys after getting their challenge ciphertexts; that is, the adversaries are not fully adaptive.

8.1 Definitions

An informal overview of our security model is as follows. The piracy adversary will be allowed to adaptively query for classical (i.e., not copy-protected) and copy-protected (i.e. quantum) functional keys. At the end of this first query phase, the adversary will produce a pair of challenge messages m^0, m^1 and $k+1$ registers (*freeloaders*) where k is the number of copy-protected keys obtained by it. After this split, the challenger presents the freeloaders each with a challenge ciphertext. Finally, after receiving the challenge ciphertexts, freeloaders can query for more functional keys, and they output their guess at the end.

We will also require the following for the challenge message pair m^0, m^1 and the functions queried. First, we require that $f(m^0) = f(m^1)$ for all functions f queried by the pirate in the *classical mode*. This is required since, otherwise, the pirate can give all the freeloaders the classical key sk_f, and they can decrypt their challenge ciphertexts with this key to distinguish $\mathsf{Enc}(m_0)$ vs $\mathsf{Enc}(m_1)$. Second, for the same reason as above, we require that a freeloader can query a key for f only if $f(m^0) = f(m^1)$. Note that these requirements are the same as the classical FE game (Definition 6). Importantly, we will not require anything for functional keys that were obtained in the *copy-protected mode* by the pirate adversary before the split. Thus, our security guarantee will allow k out of the $k+1$ freeloaders to possibly use these copy-protected functional keys to decrypt their challenge ciphertexts. However, it should not be possible for all $k+1$ registers to use these copy-protected keys simultaneously.

We also define our model so that copy-protected functional keys are generated given only a classical functional key, without any extra information. Therefore, we do not need to separately require that a copy-protected key for f allows no more than obtaining $f(m)$ given $\mathsf{Enc}(m)$, which is already implied by the regular functional encryption security.

Definition 30 (Public-key Functional Encryption with Copy-Protected Secret Keys)**.** *A public-key functional encryption scheme with copy-protected secret keys is a public-key functional encryption scheme (Definition 6) with the following additional algorithm and guarantee.*

- $\mathsf{QKeyGen}(fk)$*: Takes as input a classical functional key, outputs a quantum secret key.*

We require correctness[35] *for the quantum functional keys.*

Correctness *For all messages $m \in \mathcal{M}$,*

$$\Pr\left[\mathsf{Dec}(\mathsf{R}_{\mathsf{dec}}, ct) = f(m) : \begin{array}{c} pk, msk \leftarrow \mathsf{Setup}(1^\lambda) \\ sk_f \leftarrow \mathsf{KeyGen}(msk, f) \\ \mathsf{R}_f \leftarrow \mathsf{QKeyGen}(sk_f) \\ ct \leftarrow \mathsf{Enc}(pk, m) \end{array}\right] = 1.$$

As discussed in Section 7, correctness of the scheme along with Lemma 1 means that we can implement decryption in a way such that the quantum functional key is not disturbed. Thus, we can reuse the key to decrypt any number of times.

Similar to public-key encryption, we give a CPA-style anti-piracy security definition.

Let $\mathfrak{F} = \{\mathfrak{F}_\lambda\}_\lambda$ be a family of functions. We define anti-piracy security for $\mathfrak{F}$ as follows.

Definition 31 (CPA-Style Regular Anti-Piracy Security for Functional Encryption)**.** *Consider the following game between the challenger and an adversary $\mathcal{A}$.*

$\underline{\mathsf{FEAntiPiracy}(\lambda, \mathcal{A})}$

1. *The challenger runs $msk, pk \leftarrow \mathsf{FE.Setup}(1^\lambda)$ and submits pk to the adversary. It also initializes the set $\mathcal{F}_{clas} = \emptyset$.*

2. ***Query Phase 1:*** *For multiple rounds, the adversary adaptively submits a function $f \in \mathfrak{F}$ and a query type, either* $\mathsf{CLASSICAL}$ *or* $\mathsf{PROTECTED}$*. For each f, the challenger does the following. It first computes $sk_f \leftarrow \mathsf{FE.KeyGen}(msk, f)$.*

   Then, if the query type is $\mathsf{CLASSICAL}$*, it adds f to $\mathcal{F}_{clas}$ and submits sk_f to the adversary.*

   Otherwise, it computes $\mathsf{R}_f \leftarrow \mathsf{FE.QKeyGen}(sk_f)$ and submits R_f to the adversary.

3. ***Split Phase:*** *The adversary outputs a pair of challenge messages m^0, m^1 and a $(k+1)$-partite register $\mathsf{R}_{\mathsf{adv}}$ (where k is the number of queries of the type* $\mathsf{PROTECTED}$*), each part of the register being an interactive freeloader adversary that will be executed using a universal circuit. The challenger checks if $f(m^0) = f(m^1)$ for all $f \in \mathcal{F}_{clas}$. If not, it outputs 0 and terminates.*

[35] While our schemes satisfy perfect correctness, i.e., correctness with probability 1, some work relax the definition to $1 - \mathsf{negl}(\lambda)$.

4. ***Challenge Phase:*** *For each $\ell \in [k+1]$, the challenger samples $b_\ell \leftarrow \{0,1\}$, then computes $ct_\ell \leftarrow \mathsf{FE.Enc}(pk, m^{b_\ell})$ and sends ct_ℓ to the ℓ-th freeloader.*

5. ***Query Phase 2:*** *The challenger interacts with each of the $k+1$ freeloaders, using a universal circuit, for multiple rounds as follows. The freeloader $\ell \in [k+1]$ adaptively submits a function $f \in \mathfrak{F}$ and a query type, either $\mathsf{CLASSICAL}$ or $\mathsf{PROTECTED}$. For each query f, the challenger answers with $\perp$ if $f(m^0) \neq f(m^1)$. Otherwise, it does the following. It first computes $sk_f \leftarrow \mathsf{FE.KeyGen}(msk, f)$. If the query type is $\mathsf{CLASSICAL}$, the challenger submits sk_f to the adversary ℓ. If the query type is $\mathsf{PROTECTED}$, the challenger computes $\mathsf{R}_f \leftarrow \mathsf{FE.QKeyGen}(sk_f)$ and submits R_f to the adversary.*

6. *For $\ell \in [k+1]$, the challenger submits ct_ℓ to the ℓ-th freeloader to obtain a guess b'_ℓ. Then, it checks if $b'_\ell = b_\ell$ for all $\ell \in [k+1]$. It outputs 1 if and only if all the check pass.*

We say that a public key functional encryption scheme FE with copy-protected secret keys satisfies γ-anti-piracy security if for any QPT adversary $\mathcal{A}$,

$$\Pr[\mathsf{FEAntiPiracy}(\lambda, \mathcal{A}) = 1] \leq \frac{1}{2} + \gamma(\lambda) + \mathsf{negl}(\lambda).$$

We omit indicating γ explicitly when $\gamma = 0$.

We make some remarks about this definition. First, notice that if a construction satisfies γ-anti-piracy for any inverse polynomial γ, then it also satisfies it for $\gamma = 0$, simply because of the added $\mathsf{negl}(\lambda)$ term above. Second, note that ($\gamma = 0-$)anti-piracy security trivially implies regular functional encryption security: an adversary for the latter corresponds to an adversary for the former that only makes queries of the type $\mathsf{CLASSICAL}$.

8.2 Construction

In section, we give our construction of a functional encryption scheme with copy-protected keys for the class of functions $\mathfrak{F}$ defined as all circuits that are of size at most $Q(\lambda)$, where $Q(\lambda)$ is any fixed polynomial. The construction is highly similar to our public-key encryption construction. The main difference is that a functional key for a function f will consist of an IBE key for $id||f$ where id is a random string.

Assume the existence of following primitives where we set $\nu(\lambda) = 2^{-5\lambda - Q(\lambda)} \cdot 2^{-8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$.

- $i\mathcal{O}$, indistinguishability obfuscation scheme that is $\nu(\lambda)$-secure against $2^{5\lambda} \cdot 2^{8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$-time adversaries,

- IBE, identity-based encryption scheme with puncturable master secret keys (Definition 19) and deterministic KeyGen that satisfies strong punctured key correctness (Definition 18), for the identity space $\mathcal{ID} = \{0,1\}^{Q(\lambda)+\lambda}$ that is $\nu(\lambda)$-secure against $2^{5\lambda} \cdot 2^{8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$-time adversaries,

- F_1, puncturable PRF family with input length $Q(\lambda) + \lambda$ and output length same as the size of the randomness used by $\mathsf{CosetGen}$ (Definition 14) that is $\nu(\lambda)$-secure against $2^{5\lambda} \cdot 2^{8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$-time adversaries,

- F_2, puncturable PRF family with input length $Q(\lambda) + \lambda$ and output length same as the size of the randomness used by $\mathsf{IBE.Enc}$ that is $\nu(\lambda)$-secure against $2^{5\lambda} \cdot 2^{8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$-time adversaries,

- CCObf, compute-and-compare obfuscation for $2^{-\lambda^{0.2 \cdot C_{\mathsf{MoE.Coll}}}}$-unpredictable distributions that is $2^{-2\lambda-1} \cdot 2^{-2\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$-secure against $2^{3\lambda} \cdot 2^{2\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$-time adversaries,

Similar to our public-key encryption scheme, while we assume exponential security of the above primitives for specific exponents, these assumptions can be based only on subexponential hardness for some exponent, since we can always scale the security parameter by a polynomial factor.

Also, set $L(\lambda) = Q(\lambda) + \lambda$ and hence $c_L(\lambda) = 3 \cdot (Q(\lambda) + 2\lambda)^3$.

We now give our construction. Below, assume that all programs that are obfuscated are appropriately padded.

$\underline{\mathsf{FE.Setup}(1^\lambda)}$

1. Sample a PRF key $K_1 \leftarrow F_1.\mathsf{KeyGen}(1^\lambda)$.
2. Sample $cpk, csmk \leftarrow \mathsf{IBE.Setup}(1^\lambda)$.
3. Sample $\mathsf{OPMem} \leftarrow i\mathcal{O}(\mathsf{PMem}_{K_1})$, where PMem_{K_1} is the following program.

```
$\underline{\mathsf{PMem}_{K_1}(id||f, u_1, \ldots, u_{c_L(\lambda)}, r)}$
**Hardcoded:** $K_1$
  1. $(A_i, s_i, s'_i)_{i \in [c_L(\lambda)]} \leftarrow \mathsf{CosetGen}(1^{L(\lambda)+\lambda}; F_1(K_1, id||f))$.
  2. For each $i \in [c_L(\lambda)]$, check if $u_i \in A_i + s_i$ if $(r)_i = 0$ and check if $u_i \in A_i^\perp + s'_i$ if $(r)_i = 1$. If any of the checks fail, output 0 and terminate.
  3. Output 1.
```

4. Set $pk = (cpk, \mathsf{OPMem})$, $msk = (cmsk, K_1)$.
5. Output (pk, msk).

$\underline{\mathsf{FE.KeyGen}(msk, f)}$

1. Parse $(cmsk, K_1) = msk$.
2. Sample $id \leftarrow \{0,1\}^\lambda$.
3. Sample $ck \leftarrow \mathsf{IBE.KeyGen}(cmsk, id||f)$.
4. $(A_i, s_i, s'_i)_{i \in [c_L(\lambda)]} = \mathsf{CosetGen}(1^{L(\lambda)+\lambda}; F_1(K_1, id||f))$.
5. Output $(ck, id, f, (A_i, s_i, s'_i)_{i \in [c_L(\lambda)]})$.

$\underline{\mathsf{FE.QKeyGen}(fk)}$

1. Parse $(ck, id, f, (A_i, s_i, s'_i)_{i \in [c_L(\lambda)]}) = fk$.
2. Output $\left(\left|A_{i,s_i,s'_i}\right\rangle\right)_{i \in [c_L(\lambda)]}, ck, id, f$.

$\underline{\mathsf{FE.Enc}(pk, m)}$

1. Parse $(cpk, \mathsf{OPMem}) = pk$.
2. Sample $r \leftarrow \{0,1\}^{c_L(\lambda)}$.
3. Sample a PRF key K_2 for F_2 as $K_2 \leftarrow F_2.\mathsf{KeyGen}(1^\lambda)$.
4. Sample $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}_{\mathsf{OPMem},cpk,K_2,r,m})$, where $\mathsf{PCt}_{\mathsf{OPMem},cpk,K_2,r,m}$ is the following program.

```
$\underline{\mathsf{PCt}_{\mathsf{OPMem},cpk,K_2,r,m}(id||f, u_1, \dots, u_{c_L(\lambda)})}$
Hardcoded: $\mathsf{OPMem}, cpk, K_2, r, m$
  1. Run $\mathsf{OPMem}(id||f, u_1, \dots, u_{c_L(\lambda)}, r)$. If it outputs $0$, output $\perp$ and terminate.
  2. Output $\mathsf{IBE.Enc}(cpk, id||f, f(m); F_2(K_2, id||f))$.
```

5. Output (OPCt, r).

$\underline{\mathsf{FE.Dec}(\mathsf{R}_{\mathsf{key}}, ct)}$

1. Parse $((\mathsf{R}_i)_{i \in [c_L(\lambda)]}, ck, id, f) = \mathsf{R}_{\mathsf{key}}$ and $(\mathsf{OPCt}, r) = ct$.
2. For indices $i \in [c_L(\lambda)]$ such that $(r)_i = 1$, apply $H^{\otimes \kappa(L(\lambda)+\lambda)}$ to R_i.
3. Run the program OPCt coherently on id, f and $(\mathsf{R}_i)_{i \in [c_L(\lambda)]}$.
4. Measure the output register and denote the outcome by cct.
5. Output $\mathsf{IBE.Dec}(ck, cct)$.

Correctness with probability 1 follows in a straightforward manner from the correctness of the underlying schemes. We claim that the construction is also secure.

Theorem 33. FE *satisfies γ-anti-piracy (Definition 31) for any inverse polynomial γ.*

When we instantiate the assumed primitives with known constructions, we get the following corollary.

Corollary 7. *Assuming subexponentially secure $i\mathcal{O}$ and subexponentially secure LWE, there exists a public-key functional encryption scheme that satisfies anti-piracy security against unbounded collusion.*

8.3 Proof of Anti-Piracy

Proof will closely follow the strong anti-piracy security proof for our public-key encryption construction in Section 7.3, which crucially relies on projective implementations to simultaneously extract vectors from all registers to obtain a reduction to the monogamy-of-entanglement game. However, since the freeloaders in the functional encryption security game are interactive as opposed to the ones in regular public-key encryption, we cannot use projective implementations directly. Therefore, we first make the post-challenge-ciphertext steps non-interactive by providing the freeloaders with a punctured master secret key *pmsk* that lets them issue their own functional keys, as long as $f(m^0) = f(m^1)$.

We give the following two definitions, specific to our construction FE. Recall that we also assume that $\mathsf{IBE.KeyGen}$ is deterministic, which is true for the construction we give in Section 6.2.

We start with the post-challenge-non-interactive *regular* anti-piracy definition. It is defined similar to Definition 24.

Definition 32 (CPA-Style Post-Challenge-Ciphertext-Non-interactive Anti-Piracy Security for FE)**.** *Consider the following game between the challenger and an adversary $\mathcal{A}$.*

$\underline{\mathsf{FEAntiPiracyNI}(\lambda, \mathcal{A})}$

1. *The challenger runs $msk, pk \leftarrow \mathsf{FE.Setup}(1^\lambda)$ and submits pk to the adversary. It also initializes the set $\mathcal{F}_{clas}$. It parses $(cmsk, K_1) = msk$.*
2. ***Query Phase 1:** For multiple rounds, the adversary adaptively submits a function $f \in \mathfrak{F}$ and a query type, either $\mathsf{CLASSICAL}$ or $\mathsf{PROTECTED}$. For each f, the challenger does the following. It first computes $sk_f \leftarrow \mathsf{FE.KeyGen}(msk, f)$.*

   Then, if the query type is $\mathsf{CLASSICAL}$, it adds f to $\mathcal{F}_{clas}$ and submits sk_f to the adversary.

   Otherwise, it computes $\mathsf{R}_f \leftarrow \mathsf{FE.QKeyGen}(sk_f)$ and submits R_f to the adversary.
3. *The adversary outputs a pair of challenge messages m^0, m^1, a $(k+1)$-partite register $\mathsf{R}_{\mathsf{adv}}$ (where k is the number of queries of the type $\mathsf{PROTECTED}$) and freeloader unitaries $\{U_\ell\}_{\ell \in [k+1]}$.*
4. *The challenger checks if $f(m^0) = f(m^1)$ for all $f \in \mathcal{F}_{clas}$. If not, it outputs 0 and terminates.*

   Otherwise, the challenger computes $pmsk \leftarrow i\mathcal{O}(\mathsf{PKey}_{cmsk,K_1})$.

```
$\underline{\mathsf{PKey}_{cmsk,K_1}(id||f)}$
**Hardcoded:** $cmsk, K_1, m^0, m^1$

1. Check if $f(m^0) = f(m^1)$. If not, output $\perp$ and terminate.
2. Compute $ck = \mathsf{IBE.KeyGen}(cmsk, id||f)$.
3. $(A_i, s_i, s'_i)_{i \in [c_L(\lambda)]} = \mathsf{CosetGen}(1^{L(\lambda)+\lambda}; F_1(K_1, id||f))$.
4. Output $(ck, id, f, (A_i, s_i, s'_i)_{i \in [c_L(\lambda)]})$.
```

5. *For $\ell \in [k+1]$, the challenger executes $b'_\ell \leftarrow \mathsf{U}_{quantum}(U_\ell, \mathsf{R}_{\mathsf{adv}}[\ell], ct_\ell, pmsk)$. Then, it checks if $b'_\ell = b_\ell$ and if $f(m^0) = f(m^1)$ for all $f \in \mathcal{F}_\ell$. It outputs 1 if and only if all the check pass.*

We say FE satisfies post-challenge-ciphertext non-interactive γ-anti-piracy security if for any QPT adversary,

$$\Pr[\mathsf{FEAntiPiracyNI}(\lambda, \gamma(\lambda), \mathcal{A}) = 1] \leq \frac{1}{2} + \gamma(\lambda) + \mathsf{negl}(\lambda).$$

We now define decryptor testing and strong anti-piracy.

Definition 33 (Functional Encryption Decryptor Testing)**.** *In the anti-piracy game between the challenger and an adversary, fix $\ell \in [k+1]$, some values m^0, m^1 of the challenge messages, a freeloader unitary U_ℓ and some value st of a classical state of the challenger (which will be defined* *later). Let $\mathcal{D}$ be an efficient ciphertext and punctured master secret key distribution that can depend on st. That is, $\mathcal{D}^{st}(m;r)$ is an efficient classical algorithm where $m \in \mathcal{M}$, $r \in \mathcal{R}$ and $\mathcal{R}$ is a random coin set.*

Consider the following mixture of binary projective measurements *$\mathcal{P}$, induced by $\mathcal{D}$ and $m^0, m^1, bU_\ell st$, applied on a state ρ.*

1. *Sample $b \leftarrow \{0,1\}$.*
2. *Sample $r \leftarrow \mathcal{R}$.*
3. *Run $ct, pmsk \leftarrow \mathcal{D}^{st}(m^b; r)$.*
4. *Execute U_ℓ on $(\rho, pmsk, ct)$, and measure the first qubit of the output register, let b' be the output.*
5. *Output 1 if $b' = b$. Otherwise, output 0.*

Observe that we can efficiently execute the above measurement for arbitrary given superpositions of r and b values. Therefore, by Section 4, there exists exact and efficient approximated projective and threshold implementations for $\mathcal{P}$. We write $\mathsf{PI}_{\ell,\mathcal{D}}$ and $\mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}}$ to denote the projective implementation and approximate projective implementation of $\mathcal{P}$, respectively. Similarly, let $\mathsf{TI}_{\ell,\mathcal{D},\eta}$ and $\mathsf{ATI}^{\varepsilon,\delta}_{\ell,\mathcal{D},\eta}$ denote the threshold and efficient approximate threshold implementations of $\mathcal{P}$ for a threshold value η.

The fixed values m^0, m^1, U_ℓ, st, omitted in the notation, will be clear from the context. Unless otherwise specified, we will write $\mathcal{D}$ to denote the honest distribution where the ciphertext is sampled as

$$ct \leftarrow \mathsf{FE.Enc}(pk, m)$$

and pmsk is sampled as in Definition 32, where pk is part of st.

Definition 34 (CPA-Style Strong Anti-Piracy Security for FE)**.** *Consider the following game between the challenger and an adversary $\mathcal{A}$.*

$\mathsf{FEStrongAntiPiracy}(\lambda, \gamma, \mathcal{A})$

1. *The challenger runs $msk, pk \leftarrow \mathsf{FE.Setup}(1^\lambda)$ and submits pk to the adversary. It also initializes the set $\mathcal{F}_{clas}$. It parses $(cmsk, K_1) = msk$.*
2. ***Query Phase 1:** For multiple rounds, the adversary adaptively submits a function $f \in \mathfrak{F}$ and a query type, either* CLASSICAL *or* PROTECTED*. For each f, the challenger does the following. It first computes $sk_f \leftarrow \mathsf{FE.KeyGen}(msk, f)$.*

   Then, if the query type is CLASSICAL*, it adds f to $\mathcal{F}_{clas}$ and submits sk_f to the adversary.*

   Otherwise, it computes $\mathsf{R}_f \leftarrow \mathsf{FE.QKeyGen}(sk_f)$ and submits R_f to the adversary.
3. *The adversary outputs a pair of challenge messages m^0, m^1, a $(k+1)$-partite register $\mathsf{R}_{\mathsf{adv}}$ (where k is the number of queries of the type* PROTECTED*) and freeloader unitaries $\{U_\ell\}_{\ell \in [k+1]}$.*
4. *The challenger checks if $f(m^0) = f(m^1)$ for all $f \in \mathcal{F}_{clas}$. If not, it outputs 0 and terminates.*

5. *The challenger applies the test*

$$\bigotimes_{\ell\in[k+1]} \mathsf{TI}_{\ell,\mathcal{D},1/2+\gamma}$$

to R *and outputs* 1 *if and only if all the measurement results are* 1.

We say FE *satisfies strong* γ*-anti-piracy security if for any QPT adversary,*

$$\Pr[\mathsf{FEStrongAntiPiracy}(\lambda,\gamma(\lambda),\mathcal{A})=1]\le \mathsf{negl}(\lambda).$$

We first prove that the stronger definition implies the regular anti-piracy security (Definition 31).

Theorem 34. *Suppose* FE *satisfies strong* γ*-anti-piracy security. Then, it also satisfies regular* γ*-anti-piracy security.*

Proof. We first show that strong γ-anti-piracy security implies post-challenge-ciphertext non-interactive γ-anti-piracy security, by generalizing an argument made by [CLLZ21] for public-key encryption.

By the properties of projective implementations (Theorem 13); in the game $\mathsf{FEAntiPiracyNI}$, instead of applying the freeloader unitary and comparing the output to b'_ℓ, if we apply the corresponding projective implementation (defined in Definition 33) to obtain a value p_ℓ and output a bit $a_\ell = 1$ with probability p_ℓ, we get the correct output distribution for all registers simultaneously[36]. Hence, we can equivalently execute the security game $\mathsf{FEAntiPiracyNI}$ by applying these projective implementations, obtaining some a_ℓ, and outputting 1 if and only if $a_\ell = 1$ for all $\ell \in [k+1]$.

Now, note that by construction of TI and by the assumption that FE satisfies strong γ-anti-piracy security, we have

$$\Pr\left[\forall \ell\in[k+1]\ \ p_\ell \ge \frac{1}{2}+\gamma(\lambda)\right]\le \mathsf{negl}(\lambda).$$

Then, by above,

$$\begin{aligned}
&\Pr[\mathsf{FEAntiPiracyNI}(\lambda,\mathcal{A})=1] = \mathbb{E}[p_1\cdots p_\ell]\\
=&\Pr\left[\forall \ell\in[k+1]\ \ p_\ell \ge \frac{1}{2}+\gamma(\lambda)\right]\cdot \mathbb{E}[p_1\cdots p_\ell|\forall \ell\in[k+1]\ \ p_\ell \ge \frac{1}{2}+\gamma(\lambda)]+\\
&\Pr\left[\exists \ell\in[k+1]\ \ p_\ell < \frac{1}{2}+\gamma(\lambda)\right]\cdot \mathbb{E}[p_1\cdots p_\ell|\exists \ell\in[k+1]\ \ p_\ell < \frac{1}{2}+\gamma(\lambda)]\\
\le&\mathsf{negl}(\lambda)\cdot 1 + 1\cdot(\frac{1}{2}+\gamma(\lambda)).
\end{aligned}$$

This completes the proof that strong γ-anti-piracy security implies post-challenge-ciphertext non-interactive γ-anti-piracy security.

Now, we show that the latter implies regular γ-anti-piracy security. A freeloader adversary $\mathcal{B}'$ for $\mathsf{FEAntiPiracyNI}$ can simulate a freeloader adversary $\mathcal{B}$ for the regular γ-anti-piracy as follows. Whenever $\mathcal{B}$ makes a query for a function f in Query Phase 2, $\mathcal{B}'$ samples a random identity id and evaluates $pmsk$ on id, f. Since f satisfies $f(m^0) = f(m^1)$, by correctness of $i\mathcal{O}$, it will be able to obtain the correct key. If query is of type $\mathsf{CLASSICAL}$, then $\mathcal{B}'$ submits the obtained classical key fk to $\mathcal{B}$. Otherwise, it runs $\mathsf{FE.QKeyGen}$ on fk and then submits the resulting quantum key. This perfectly simulates the regular anti-piracy game, hence post-challenge-ciphertext non-interactive γ-anti-piracy security implies regular anti-piracy security. □

[36] When used directly, Theorem 13 would give us this for only single register at a time. However, note that the joint distribution is also correct since the projective implementations are correct for any input state, and hence we can consider the post-measurement state of any register conditioned on measurement outcomes of the other registers, and the projective implementation will still have the correct output distribution.

Reducing to Monogamy-of-Entanglement

Theorem 35. FE *satisfies strong γ-anti-piracy security for any inverse polynomial γ.*

Combining this theorem with the results from the previous section yields that our FE construction is secure. Proof of this theorem is almost identical to the proof of security of our public-key encryption scheme (Section 7.3). Therefore, we will omit the proofs of some sub-claims.

Throughout the proof, we will interpret identity strings for IBE, which are $Q(\lambda) + \lambda$-bit strings, as integers in the set $\{0, 1, \dots, 2^{Q(\lambda)+\lambda} - 1\}$.

Fix any inverse polynomial $\gamma(\lambda)$ and suppose for a contradiction that there exists an efficient adversary $\mathcal{A}$ that wins the strong γ-anti-piracy game with non-negligible probability. Let k denote the number of copy-protected keys obtained by the adversary. Define Hyb_0 to be the original security game $\mathsf{FEStrongAntiPiracy}(\lambda, \gamma(\lambda), \mathcal{A})$.

Define Hyb_1 by modifying Hyb_0 as follows. When generating functional keys, we sample the random identity strings id in a collision-free way. Further, at the end of the game, the challenger instead applies the test $\bigotimes_{\ell \in [k+1]} \mathsf{ATI}^{\varepsilon,\delta}_{\ell,\mathcal{D},1/2+\frac{31\gamma}{32}}$ instead of TI where we set $\varepsilon = \frac{\gamma}{32k}$ and $\delta = 2^{-10\lambda} \cdot 2^{-10\lambda^{C_{\mathsf{MoE.Coll}}}}$.

Claim 17. $\Pr[\mathsf{Hyb}_2 = 1] > 1/p(\lambda)$ *for some polynomial $p(\cdot)$ and infinitely many values of $\lambda > 0$*

Proof. Follows from the same argument as in Section 7.3. □

Definition 35. *For all $j \in [k]$, let $id_j || f_j$ denote the identity string and let $(A^j_i, s^j_i, s'^j_i)_{i \in [c_L(\lambda)]}$ denote the tuple of cosets sampled during the sampling of the functional key for the $(\alpha^{-1}(j))$-th query of type* $\mathsf{PROTECTED}$*. That is, it is the coset tuple associated with $id_j || f_j$.*

We now define a monogamy-of-entanglement type game $\mathcal{G}$, similar to the game defined in the PKE proof.

$\underline{\mathcal{G}(\lambda, (\mathcal{A}'_0, \mathcal{A}'_1, \mathcal{A}'_2))}$

1. The challenger runs $msk, pk \leftarrow \mathsf{FE.Setup}(1^\lambda)$ and submits pk to the adversary. It also initializes the set $\mathcal{F}_{clas}$. It parses $(cmsk, K_1) = msk$.

2. **Query Phase 1:** For multiple rounds, the adversary adaptively submits a function $f \in \mathfrak{F}$ and a query type, either $\mathsf{CLASSICAL}$ or $\mathsf{PROTECTED}$. For each f, the challenger does the following. It first computes $sk_f \leftarrow \mathsf{FE.KeyGen}(msk, f)$.

   Then, if the query type is $\mathsf{CLASSICAL}$, it adds f to $\mathcal{F}_{clas}$ and submits sk_f to the adversary.

   Otherwise, it computes $\mathsf{R}_f \leftarrow \mathsf{FE.QKeyGen}(sk_f)$ and submits R_f to the adversary.

3. The adversary outputs a pair of challenge messages m^0, m^1 and an index $j^* \in [k]$ where k is the number of queries it made of type $\mathsf{PROTECTED}$.

4. The challenger checks if $f(m^0) = f(m^1)$ for all $f \in \mathcal{F}_{clas}$. If not, it outputs 0 and terminates.

   Otherwise, the challenger computes $K_1\{id_{j^*} || f_{j^*}\} \leftarrow F_1.\mathsf{Punc}(K_1, id_{j^*} || f_{j^*})$ and submits it to the adversary.

5. The challenger outputs a *bipartite* register $\mathsf{R}_{\mathsf{bip}}$.

6. For $\ell \in \{1, 2\}$, the challenger does the following.

6.1. Sample $r_\ell \leftarrow \{0,1\}^{c_L(\lambda)}$.

6.2. Run $\mathcal{A}'_\ell$ on $\mathsf{R}_{\mathsf{bip}}[\ell]$, $(A_i^{j^*})_{i\in[c_L(\lambda)]}$ and r_ℓ to obtain a tuple of vectors $(v_{\ell,i})_{i\in[c_L(\lambda)]}$.

6.3. For all $i \in [c_L(\lambda)]$, check if $v_{\ell,i} \in A_i^{j^*} + s_i^{j^*}$ if $(r_\ell)_i = 0$ and check if $v_{\ell,i} \in (A^{j^*})_i^\perp + s_i'^{j^*}$ if $(r_\ell)_i = 1$.

If all the checks pass, the challenger outputs 1. Otherwise, it outputs 0.

It is straightforward to reduce this game to the collusion-resistant MoE game. We construct our adversary for $\mathcal{G}$ as follows, where we will define challenge ciphertext-punctured master secret key distributions $\mathcal{D}_j$ later. Without loss of generality[37], we will assume that all the queries made by the adversary $\mathcal{A}$ of type $\mathsf{PROTECTED}$ satisfy $f(m_0) \neq f(m_1)$.

$\underline{\mathcal{A}'_0(pk)}$

1. Simulate $\mathcal{A}$ on pk by making a functional key query to the challenger whenever $\mathcal{A}$ makes a query, and forwarding the obtained key to it. Let $\mathsf{R}_{\mathsf{adv}}$ be the $(k+1)$-partite register (with state σ) and (m^0, m^1) be the challenge messages output by $\mathcal{A}$ at the end of the query phase.
2. Uniformly at random sample x, y, j^* such that $1 \le x < y \le k+1$ and $j^* \in \{1, \dots, k\}$.
3. Output j^* to the challenger and obtain $K_1\{id_{j^*}||f_{j^*}\}$.
4. Apply $\mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}_0}$ to all registers $\mathsf{R}_{\mathsf{adv}}[\ell]$ for $\ell \in [k+1]$, let $b_{\ell,0}$ be the measurement outcomes.
5. Apply $\mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}_i}$ in succession for $i = 1$ to j^* to $\mathsf{R}_{\mathsf{adv}}[x]$, let $b_{x,i}$ be the measurement outcomes.
6. Apply $\mathsf{API}^{\varepsilon,\delta}_{\ell,\mathcal{D}_i}$ in succession for $i = 1$ to j^* to $\mathsf{R}_{\mathsf{adv}}[y]$, let $b_{y,i}$ be the measurement outcomes.
7. Output

$$\begin{aligned}
&((\mathsf{R}_{\mathsf{adv}}[x], j^*, x, y, (b_{\ell,0})_{\ell\in[k+1]}, (b_{x,i})_{i\in[j^*]}, (b_{y,i})_{i\in[j^*]}),\\
&(\mathsf{R}_{\mathsf{adv}}[y], j^*, x, y, (b_{\ell,0})_{\ell\in[k+1]}, (b_{x,i})_{i\in[j^*]}, (b_{y,i})_{i\in[j^*]},),\\
&j^*).
\end{aligned}$$

For $j \in \{1, \dots, k\}$, define $\mathcal{D}_j$ to be the following challenge ciphertext-punctured master secret key distribution.

1. Sample $r \leftarrow \{0,1\}^{c_L(\lambda)}$.
2. Sample a PRF key K_2 for F_2 as $K_2 \leftarrow F_2.\mathsf{KeyGen}(1^\lambda)$.
3. Sample $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}^{(j)}_{\mathsf{OPMem},cpk,K_2,r,m,id_j||f_j})$

[37] In the general case, the adversary would simply sample j^* from $[k']$ where k' is the number of queries made by the adversary $\mathcal{A}$ of type $\mathsf{PROTECTED}$ that do satisfy $f(m_0) \neq f(m_1)$.

```
$\mathsf{PCt}^{(j)}_{\mathsf{OPMem},cpk,K_2,r,m^b,m_{1-b},id_j||f_j}(id||f,u_1,\ldots,u_{c_L(\lambda)})$

Hardcoded: $\mathsf{OPMem}, cpk, K_2, r, m^b, m^{1-b}, id_j||f_j$

  1. Run $\mathsf{OPMem}(id||f,u_1,\ldots,u_{c_L(\lambda)},r)$. If it outputs $0$, output $\perp$ and terminate.
  2. If $id||f < id_j||f_j$, set $a = f(m^{1-b})$. Otherwise, set $a = f(m^b)$.
  3. Output $\mathsf{IBE.Enc}(cpk, id, a; F_2(K_2, id))$.
```

4. Sample $pmsk \leftarrow i\mathcal{O}(\mathsf{PKey}_{cmsk,K_1\{id_{j^*}||f_{j^*}\}})$.

```
$\mathsf{PKey}_{cmsk,K_1\{id_{j^*}||f_{j^*}\}}(id||f)$

Hardcoded: $cmsk, K_1\{id_{j^*}||f_{j^*}\}, m^0, m^1$

  1. Check if $f(m^0) = f(m^1)$. If not, output $\perp$ and terminate.
  2. Compute $ck = \mathsf{IBE.KeyGen}(cmsk, id||f)$.
  3. $(A_i, s_i, s'_i)_{i\in[c_L(\lambda)]} = \mathsf{CosetGen}(1^{L(\lambda)+\lambda}; F_1(K_1\{id_{j^*}||f_{j^*}\}, id||f))$.
  4. Output $(ck, id, f, (A_i, s_i, s'_i)_{i\in[c_L(\lambda)]})$.
```

5. Output $(\mathsf{OPCt}, r), pmsk$.

We define $\mathcal{D}_0$ to be the distribution where ciphertext is computed honestly and $pmsk$ is computed as in Definition 32. We define $\mathcal{D}_{k+1}$ as follows.

1. Sample $r \leftarrow \{0,1\}^{c_L(\lambda)}$.

2. Sample a PRF key K_2 for $F_2.\mathsf{KeyGen}(1^\lambda)$.

3. Sample $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}^{(k+1)}_{\mathsf{OPMem},cpk,K_2,r})$

```
$\mathsf{PCt}^{(k+1)}_{\mathsf{OPMem},cpk,K_2,r,m^0,m^1,b}(id||f,u_1,\ldots,u_{c_L(\lambda)})$

Hardcoded: $\mathsf{OPMem}, cpk, K_2, r, m^{1-b}$

  1. Run $\mathsf{OPMem}(id||f,u_1,\ldots,u_{c_L(\lambda)},r)$. If it outputs $0$, output $\perp$ and terminate.
  2. Output $\mathsf{IBE.Enc}(cpk, id, f(m^{1-b}); F_2(K_2, id))$.
```

4. Sample $pmsk$ as in $\mathcal{D}_j$.

5. Output $(\mathsf{OPCt}, r), pmsk$.

Definition 36. *Let $\mathsf{Exp}_{\mathcal{C},\ell}$ denote the outcome of the following experiment where $\mathcal{C}$ is a challenge ciphertext-punctured master secret key distribution that can depend on pp.*

1. *Execute $pk, sk \leftarrow \mathsf{PKE.Setup}(1^\lambda)$.*

2. *Simulate the first steps of $\mathcal{A}'_0$ and the challenger of $\mathcal{G}$:*

   2.1. *Simulate $\mathcal{A}$ on pk by making a functional key query to the challenger whenever $\mathcal{A}$ makes a query, and forwarding the obtained key to it. Let $\mathsf{R}_{\mathsf{adv}}$ be the $(k+1)$-partite register (with state σ) and (m^0, m^1) be the challenge messages output by $\mathcal{A}$ at the end of the query phase.*
   2.2. *Uniformly at random sample x, y, j^* such that $1 \leq x < y \leq k+1$ and $j^* \in \{1, \dots, k\}$.*
   2.3. *Compute $K_1\{id_{j^*}||f_{j^*}\} \leftarrow F_1.\mathsf{Punc}(K_1, id_{j^*}||f_{j^*})$.*

3. *Set $pp = (x, y, j^*, (id_j||f_j)_{j \in [k+1]}, m_0, m_1, pk)$.*

4. *Sample $b \leftarrow \{0,1\}$.*

5. *Sample $ct, pmsk \leftarrow \mathcal{C}(pp, m^b)$.*

6. *Output $\mathsf{R}_{\mathsf{adv}}, (b, ct, pmsk), pp$.*

We will write $\mathsf{Exp}_{\mathcal{C},\ell} \approx^c_\nu \mathsf{Exp}_{\mathcal{C}',\ell}$ to denote that the advantage of any computational adversary in distinguishing the outcomes of these experiments is ν.

Claim 18. *Let τ be the state of the bipartite register $\mathsf{R}_{\mathsf{adv}}[x,y]$ output by $\mathcal{A}'_0$ in $\mathcal{G}$, and also consider the classical values $j^*, x, y, \{b_{\ell,i}\}_{\ell,i}$ contained in the output of $\mathcal{A}'_0$.*

Suppose we apply the measurement $\mathsf{API}^{\varepsilon,\delta}_{x,\mathcal{D}_{j^*+1}} \otimes \mathsf{API}^{\varepsilon,\delta}_{y,\mathcal{D}_{j^*+1}}$ to τ and let b_{x,j^*+1}, b_{y,j^*+1} denote the measurement outcomes we obtain. Then,

$$\Pr\left[b_{x,j^*} - b_{x,j^*+1} > \frac{29\gamma}{32k} \wedge b_{y,j^*} - b_{y,j^*+1} > \frac{29\gamma}{32k}\right] > \frac{1}{4p(\lambda) \cdot k^3(\lambda)}$$

where the probability is taken over the randomness of the challenger, the adversary $\mathcal{A}'_0$ and the measurement outcomes.

Proof. Follows from the same argument as Claim 5. Only caveat is that we need to prove that the success probability of the freeloaders with respect to the challenge ciphertext distribution $\mathcal{D}_{k+1}$ is $\leq 1/2$. However, this is indeed true, since $\mathcal{D}_{k+1}$ is encoding m^{1-b} while the challenge bit is b. □

Definition 37. *When we refer to $((id||f) + \Delta)$ as a function, we mean the following. We associate the strings in $\{0,1\}^{\lambda+Q(\lambda)}$ with numbers $\{0, 1, \dots, 2^{\lambda+Q(\lambda)}\}$ in the canonical way, and we compute the sum of the numbers associated with Δ and $(id||f)$. Then, we switch back to the bit representation of this number, and take the last Q bits, and we define $(id||f) + \Delta$ to be the circuit defined by this Q-bit string.*

Now, we define some intermediary distributions. Define the following for all $j \in \{0, 1, \dots, k\}$ and $\Delta \in \{0, 1, \dots, id_{j+1}||f_{j+1} - id_j||f_j - 1\}$. For notational convenience, also define $\mathcal{D}_j^{id_{j+1}||f_{j+1} - id_j||f_j, 0}$ to be $\mathcal{D}_{j+1}^{(0,0)}$ for all $j \in \{0, 1, \dots, k\}$. Also note that $\mathcal{D}_j^{(0,0)}$ is exactly the same as $\mathcal{D}_j$ for $j \in [k]$.

- $\underline{\mathcal{D}_j^{(\Delta,0)}}$:

  1. Sample $r \leftarrow \{0,1\}^{c_L(\lambda)}$.
  2. Sample a PRF key K_2 for $F_2.\mathsf{KeyGen}(1^\lambda)$.
  3. Sample $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}^{(j,\Delta,0)}_{\mathsf{OPMem},cpk,K_2,r,m,id_j+\Delta})$.

$\underline{\mathsf{PCt}^{(j,\Delta,0)}_{\mathsf{OPMem},cpk,K_2,r,m^b,m^{1-b},id_j||f_j+\Delta}(id||f,u_1,\ldots,u_{c_L(\lambda)})}$

Hardcoded: $\mathsf{OPMem}, cpk, K_2, r, m^b, m^{1-b}, id_j||f_j+\Delta$

(a) Run $\mathsf{OPMem}(id||f,u_1,\ldots,u_{c_L(\lambda)},r)$. If it outputs 0, output $\perp$ and terminate.

(b) If $id||f < id_j||f_j + \Delta$, set $a = f(m^{1-b})$. Otherwise, set $a = f(m^b)$.

(c) Output $\mathsf{IBE.Enc}(cpk, id, a; F_2(K_2, id))$.

4. Sample $pmsk$ as in $\mathcal{D}_j$.
5. Output $(\mathsf{OPCt}, r), pmsk$.

- $\underline{\mathcal{D}_j^{(\Delta,1)}}$:

1. Sample $r \leftarrow \{0,1\}^{c_L(\lambda)}$.
2. Sample a PRF key K_2 for $F_2.\mathsf{KeyGen}(1^\lambda)$.
3. $ct^* = \mathsf{IBE.Enc}(cpk, id_j||f_j+\Delta, (id_j||f_j+\Delta)(m^b); F_2(K_2, id_j||f_j+\Delta))$.
4. $K_2\{id_j||f_j+\Delta\} \leftarrow F_2.\mathsf{Punc}(K_2, id_j||f_j+\Delta)$.
5. Sample $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}^{(j,\Delta,1)}_{\mathsf{OPMem},cpk,K_2\{id_j+\Delta\},r,m,id_j+\Delta,ct^*})$.

$\underline{\mathsf{PCt}^{(j,\Delta,1)}_{\mathsf{OPMem},cpk,K_2\{id_j+\Delta\},r,m,id_j+\Delta,ct^*}(id||f,u_1,\ldots,u_{c_L(\lambda)})}$

Hardcoded: $\mathsf{OPMem}, cpk, K_2\{id_j||f_j+\Delta\}, r, m^b, m^{1-b}, id_j||f_j+\Delta, ct^*$

(a) Run $\mathsf{OPMem}(id||f,u_1,\ldots,u_{c_L(\lambda)},r)$. If it outputs 0, output $\perp$ and terminate.

(b) If $id||f = id_j||f_j + \Delta$, output ct^* and terminate.

(c) If $id < id_j||f_j + \Delta + 1$, set $a = f(m^{1-b})$. Otherwise, set $a = f(m)$.

(d) Output $\mathsf{IBE.Enc}(cpk, id, a; F_2(K_2, id))$.

6. Sample $pmsk$ as in $\mathcal{D}_j$.
7. Output $(\mathsf{OPCt}, r), pmsk$.

- $\underline{\mathcal{D}_j^{(\Delta,2)}}$:

1. Sample $r \leftarrow \{0,1\}^{c_L(\lambda)}$.
2. Sample a PRF key K_2 for $F_2.\mathsf{KeyGen}(1^\lambda)$.
3. Sample z^* uniformly at random from the output space of F_2.
4. $ct^* = \mathsf{IBE.Enc}(cpk, id_j||f_j+\Delta, (id_j||f_j+\Delta)(m^b); z^*)$.
5. $K_2\{id_j||f_j+\Delta\} \leftarrow F_2.\mathsf{Punc}(K_2, id_j||f_j+\Delta)$.
6. Sample $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}^{(j,\Delta,2)}_{\mathsf{OPMem},cpk,K_2\{id_j+\Delta\},r,m,id_j+\Delta,ct^*})$.

$\underline{\mathsf{PCt}^{(j,\Delta,2)}_{\mathsf{OPMem},cpk,K_2\{id_j||f_j+\Delta\},r,m^b,m^{1-b},id_j||f_j+\Delta,ct^*}(id||f,u_1,\ldots,u_{c_L(\lambda)})}$

Hardcoded: $\mathsf{OPMem}, cpk, K_2\{id_j+\Delta\}, r, m^b, m^{1-b}, id_j||f_j+\Delta, ct^*$

(a) Run $\mathsf{OPMem}(id||f,u_1,\ldots,u_{c_L(\lambda)},r)$. If it outputs 0, output $\perp$ and terminate.

(b) If $id||f = id_j||f_j + \Delta$, output ct^* and terminate.

(c) If $id < id_j||f_j + \Delta + 1$, set $a = f(m^{1-b})$. Otherwise, set $a = f(m)$.

```
(d) Output $\mathsf{IBE.Enc}(cpk, id, a; F_2(K_2, id))$.
```

7. Sample $pmsk$ as in $\mathcal{D}_j$.
8. Output $(\mathsf{OPCt}, r), pmsk$.

- $\underline{\mathcal{D}_j^{(\Delta,3)}}$:

1. Sample $r \leftarrow \{0,1\}^{c_L(\lambda)}$.
2. Sample a PRF key K_2 for $F_2.\mathsf{KeyGen}(1^\lambda)$.
3. Sample z^* uniformly at random from the output space of F_2.
4. $ct^* = \mathsf{IBE.Enc}(cpk, id_j||f_j + \Delta, (id_j||f_j + \Delta)(m^b); z^*)$.
5. $K_2\{id_j||f_j + \Delta\} \leftarrow F_2.\mathsf{Punc}(K_2, id_j||f_j + \Delta)$.
6. Compute $(A_i^*, s_i^*, s_i^{\prime *}) = F_1(K_1, id_j||f_j + \Delta)$.
7. For $i \in [c_L(\lambda)]$, set $g_i = \mathsf{Can}_{A_i^*}$ if $(r)_i = 0$ and set $g_i = \mathsf{Can}_{(A_i^*)^\perp}$ if $(r)_i = 1$.
8. For $i \in [c_L(\lambda)]$, compute $y_i = g_i(s_i^*)$ if $(r)_i = 0$ and $y_i = g_i(s_i^{\prime *})$ if $(r)_i = 1$.
9. Set g to be the function $g(v_1, \ldots, v_{c_L(\lambda)}) = (g_1(v_1)|| \ldots ||g_{c_L(\lambda)}(v_{c_L(\lambda)}))$.
10. Set $y = y_1|| \ldots ||y_{c_L(\lambda)}$.
11. $\mathsf{OCC} \leftarrow \mathsf{CCObf.Obf}(g, y, ct^*)$.
12. Sample $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}^{(j,\Delta,3)}_{\mathsf{OPMem},cpk,K_2\{id_j\},r,m,id_j+\Delta,\mathsf{OCC}})$.

```
$\underline{\mathsf{PCt}^{(j,\Delta,3)}_{\mathsf{OPMem},cpk,K_2\{id_j+\Delta\},r,m,id_j||f_j+\Delta,\mathsf{OCC}}(id||f, u_1, \ldots, u_{c_L(\lambda)})}$
Hardcoded: $\mathsf{OPMem}, cpk, K_2\{id_j + \Delta\}, r, m^b, m^{1-b}, id_j||f_j + \Delta, \mathsf{OCC}$
(a) If $id||f = id_j||f_j + \Delta$, output the output of $\mathsf{OCC}(u_1, \ldots, u_{c_L(\lambda)})$ and terminate.
(b) Run $\mathsf{OPMem}(id||f, u_1, \ldots, u_{c_L(\lambda)}, r)$. If it outputs 0, output $\perp$ and terminate.
(c) If $id < id_j||f_j + \Delta + 1$, set $a = f(m^{1-b})$. Otherwise, set $a = f(m)$.
(d) Output $\mathsf{IBE.Enc}(cpk, id, a; F_2(K_2, id))$.
```

13. Sample $pmsk$ as in $\mathcal{D}_j$.
14. Output $(\mathsf{OPCt}, r), pmsk$.

- $\underline{\mathcal{D}_j^{(\Delta,4)}}$: Same as $\mathcal{D}_j^{(\Delta,3)}$ except for the following. Replace the line

$$ct^* = \mathsf{IBE.Enc}(cpk, id_j||f_j + \Delta, (id_j||f_j + \Delta)(m^b); z^*)$$

with

$$ct^* = \mathsf{IBE.Enc}(cpk, id_j||f_j + \Delta, (id_j||f_j + \Delta)(m^{1-b}); z^*).$$

- $\underline{\mathcal{D}_j^{(\Delta,5)}}$: Same as $\mathcal{D}_j^{(\Delta,2)}$ except for the following. Replace the line

$$ct^* = \mathsf{IBE.Enc}(cpk, id_j||f_j + \Delta, (id_j||f_j + \Delta)(m^b); z^*)$$

with

$$ct^* = \mathsf{IBE.Enc}(cpk, id_j||f_j + \Delta, (id_j||f_j + \Delta)(m^{1-b}); z^*).$$

- $\underline{\mathcal{D}_j^{(\Delta,6)}}$: Same as $\mathcal{D}_j^{(\Delta,1)}$ except for the following. Replace the line

  $$ct^* = \mathsf{IBE.Enc}(cpk, id_j||f_j + \Delta, (id_j||f_j + \Delta)(m^b); F_2(K_2, id_j||f_j + \Delta))$$

  with

  $$ct^* = \mathsf{IBE.Enc}(cpk, id_j||f_j + \Delta, {\color{red}(id_j||f_j + \Delta)(m^{1-b})}; F_2(K_2, id_j||f_j + \Delta)).$$

Now, we show that these distributions *collapse* around $\Delta = 0$ for each j. Below, all our indistinguishability claims are for $2^{5\lambda} \cdot 2^{8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$-time adversaries and we set $\nu(\lambda) = 2^{-5\lambda - Q(\lambda)} \cdot 2^{-8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$.

Claim 19. $\mathsf{Exp}_{\mathcal{D}_j^{(\Delta,0)},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(\Delta,1)},\ell}$ *for all* $j \in \{0, 1, \dots, k\}$*,* $\Delta \in \{0, 1, \dots, id_{j+1} - id_j - 1\}$ *and* $\ell \in [k+1]$.

Proof. Observe that by punctured key correctness of F_2 (Definition 1), the different obfuscated programs $\mathsf{PCt}^{(j,\Delta,0)}_{\mathsf{OPMem},cpk,K_2,r,m,id_j+\Delta}$ and $\mathsf{PCt}^{(j,\Delta,1)}_{\mathsf{OPMem},cpk,K_2\{id_j+\Delta\},r,m,id_j+\Delta,ct^*}$ in these hybrids have the same functionality. The result follows by security of $i\mathcal{O}$ and by our choice of parameters. □

Claim 20. $\mathsf{Exp}_{\mathcal{D}_j^{(\Delta,1)},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(\Delta,2)},\ell}$ *for all* $j \in \{0, 1, \dots, k\}$*,* $\Delta \in \{0, 1, \dots, id_{j+1} - id_j - 1\}$ *and* $\ell \in [k+1]$.

Proof. The result follows by selective puncturing security of F_2 (Definition 1) and our choice of parameters. □

Claim 21. $\mathsf{Exp}_{\mathcal{D}_j^{(\Delta,2)},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(\Delta,3)},\ell}$ *for all* $j \in \{0, 1, \dots, k\}$*,* $\Delta \in \{0, 1, \dots, id_{j+1} - id_j - 1\}$ *and* $\ell \in [k+1]$.

Proof. Observe that the obfuscated ciphertext programs PCt in these hybrids have the same functionality by correctness of CCObf, since a vector w is in $A_i^* + s_i^*$ if and only if $\mathsf{Can}_{A_i^*}(w) = \mathsf{Can}_{A_i^*}(s_i^*)$ and similarly for $(A^*)_i^{\perp} + s_i^{'*}$. Then, the claim follows by the security of $i\mathcal{O}$. □

Claim 22. $\mathsf{Exp}_{\mathcal{D}_j^{(\Delta,3)},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(\Delta,4)},\ell}$ *if*

- $j \in \{1, \dots, k\}$ *and* $\Delta \in \{1, \dots, id_{j+1} - id_j - 1\}$*, or*
- $j = 0$ *and* $\Delta \in \{0, 1, \dots, id_{j+1} - id_j - 1\}$

and for all $\ell \in [k+1]$.

Proof. We have two cases. First, assume that $(id_j||f_j + \Delta)(m_0) = (id_j||f_j + \Delta)(m_1)$. Then, the result easily follows.

Otherwise, define the intermediary distributions $\mathcal{D}_j^{(\Delta,3')}, \mathcal{D}_j^{(\Delta,4')}$ as follows.

$\underline{\mathcal{D}_j^{(\Delta,3')}}$

1. Sample (OPCt, r) as in $\mathcal{D}_j^{(\Delta,3)}$.
2. Sample $cmsk' \leftarrow \mathsf{IBE.Punc}(cmsk, id_j||f_j + \Delta)$.
3. Sample $pmsk \leftarrow i\mathcal{O}(\mathsf{PKey}_{cmsk',K_1\{id_{j^*}||f_{j^*}\}})$.

```
$\mathsf{PKey}_{cmsk',K_1\{id_{j^*}||f_{j^*}\}}(id||f)$
**Hardcoded:** $cmsk', K_1\{id_{j^*}||f_{j^*}\}, m^0, m^1$
  1. Check if $f(m^0) = f(m^1)$. If not, output $\perp$ and terminate.
  2. Compute $ck = \mathsf{IBE.KeyGen}(cmsk', id||f)$.
  3. $(A_i, s_i, s'_i)_{i\in[c_L(\lambda)]} = \mathsf{CosetGen}(1^{L(\lambda)+\lambda}; F_1(K_1\{id_{j^*}||f_{j^*}\}, id||f))$.
  4. Output $(ck, id, f, (A_i, s_i, s'_i)_{i\in[c_L(\lambda)]})$.
```

$\underline{\mathcal{D}_j^{(\Delta,4')}}$ Same as $\mathcal{D}_j^{(\Delta,3')}$ except for the following. Replace the line

$$ct^* = \mathsf{IBE.Enc}(cpk, id_j||f_j + \Delta, (id_j||f_j + \Delta)(m^b); z^*)$$

with

$$ct^* = \mathsf{IBE.Enc}(cpk, id_j||f_j + \Delta, (id_j||f_j + \Delta)(m^{1-b}); z^*).$$

First, we claim $\mathsf{Exp}_{\mathcal{D}_j^{(\Delta,3)},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(\Delta,3')},\ell}$ and $\mathsf{Exp}_{\mathcal{D}_j^{(\Delta,4)},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(\Delta,4')},\ell}$. We will only argue the first one and the second one follows similarly. Observe that by strong punctured key correctness of deterministic $\mathsf{IBE.KeyGen}$, the obfuscated programs PKey in these hybrids can behave differently only on input $(id_j||f_j + \Delta)(m_0)$.. However, since we are considering the case $(id_j||f_j + \Delta)(m_0) \neq (id_j||f_j + \Delta)(m_1)$, the programs will not go past the first line and will have the exact same functionality. Then, the claim follows by security of $i\mathcal{O}$.

Now, we claim $\mathsf{Exp}_{\mathcal{D}_j^{(\Delta,3')},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(\Delta,4')},\ell}$. Observe that in these hybrids, the randomness used to invoke $\mathsf{IBE.Enc}$ to compute ct^* is uniformly and independently sampled. Further, the adversary only has the IBE keys for the identities $id_1||f_1, id_2||f_2, \dots, id_k||f_k$, all of which are different from the identity $id_j||f_j + \Delta$ under which ct^* is encrypted. Finally, the master secret key $cmsk'$ obtained by the adversary is punctured at $id_j||f_j + \Delta$. Hence, the result follows from the punctured master secret key security of IBE. □

Claim 23. *$\mathsf{Exp}_{\mathcal{D}_j^{(\Delta,4)},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(\Delta,5)},\ell}$ for all $j \in \{0, 1, \dots, k\}$, $\Delta \in \{0, 1, \dots, id_{j+1} - id_j - 1\}$ and $\ell \in [k + 1]$.*

Proof. Essentially the same argument as in Claim 21 yields the result. □

Claim 24. *$\mathsf{Exp}_{\mathcal{D}_j^{(\Delta,5)},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(\Delta,6)},\ell}$ for all $j \in \{0, 1, \dots, k\}$, $\Delta \in \{0, 1, \dots, id_{j+1} - id_j - 1\}$ and $\ell \in [k + 1]$.*

Proof. Essentially the same argument as in Claim 20 yields the result. □

Claim 25. *$\mathsf{Exp}_{\mathcal{D}_j^{(\Delta,6)},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(\Delta+1,0)},\ell}$ for all $j \in \{0, 1, \dots, k\}$, $\Delta \in \{0, 1, \dots, id_{j+1} - id_j - 1\}$ and $\ell \in [k + 1]$.*

Proof. Essentially the same argument as in Claim 19 yields the result. □

Claim 26. *For all $\ell \in [k + 1]$, we have*

- $\mathsf{Exp}_{\mathcal{D}_0,\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_1,\ell}$
- $\mathsf{Exp}_{\mathcal{D}_j^{(0,4)},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_{j+1},\ell}$ *for all $j \in \{0, 1, \dots, k\}$*

- $\mathsf{Exp}_{\mathcal{D}_j,\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(0,3)},\ell}$ *for all* $j \in \{0, 1, \ldots, k\}$

where $\nu(\lambda) = 2^{-5\lambda} \cdot 2^{-8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$.

Proof. It is easy to see that $\mathcal{D}_0 \approx^c_{\nu(\lambda)} \mathcal{D}_0^{(0,0)}$ and $\mathcal{D}_{k+1} \approx^c_{\nu(\lambda)} \mathcal{D}_{k+1}^{(0,0)}$ by the security of $i\mathcal{O}$. For the former, in particular, observe that the obfuscated punctured master secret key programs have the same functionality since while K_1 in $\mathcal{D}_0$ is punctured at $id_{j^*}||f_{j^*}$, we have that $f_{j^*}(m_0) \neq f_{j^*}(m_1)$.

Rest follows by a simple calculation using the above results. □

Definition 38. *We will write* $\mathcal{D}'$ *to denote* $\mathcal{D}_{j^*}^{(0,3)}$ *and* $\mathcal{D}''$ *to denote* $\mathcal{D}_{j^*}^{(0,4)}$ *where* j^* *is as output by* $\mathcal{A}'_0$.

Claim 27. *Let* τ *be the bipartite state output by* $\mathcal{A}'_0$ *in* $\mathcal{G}$. *Let* p'_x, p'_y *be the outcome of applying* $\mathsf{PI}_{x,\mathcal{D}'} \otimes \mathsf{PI}_{y,\mathcal{D}'}$ *to* τ. *Similarly, let* p''_x, p''_y *be the outcome of applying* $\mathsf{PI}_{x,\mathcal{D}''} \otimes \mathsf{PI}_{y,\mathcal{D}''}$ *to* τ. *Then,*

- $\Pr\Big[p'_x > b_{x,j^*} - \frac{3\gamma}{32k} \wedge p'_y > b_{y,j^*} - \frac{3\gamma}{32k}\Big] \geq 1 - 2^{-2\lambda} \cdot 2^{-4\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$.

- $\Pr\Big[b_{x,j^*} - p''_x > \frac{28\gamma}{32k} \wedge b_{y,j^*} - p''_y > \frac{28\gamma}{32k}\Big] > \frac{1}{q(\lambda)}$ *for some polynomial* $q(\cdot)$.

Proof. Follows from the same argument as in Claim 14. □

Claim 28. *There exist efficient* $\mathcal{A}'_1, \mathcal{A}'_2$ *such that* $(\mathcal{A}'_0, \mathcal{A}'_1, \mathcal{A}'_2)$ *wins* $\mathcal{G}$ *with probability* $\frac{1}{2^{0.4\cdot\lambda^{C_{\mathsf{MoE.Coll}}}}}$.

Proof. Follows from the same argument as in Claim 15. □

Claim 29. *There exists efficient* $\mathcal{A}'' = (\mathcal{A}''_0, \mathcal{A}''_1, \mathcal{A}''_2)$ *such that*

$$\Pr\big[\mathsf{Moe-Coll-PuncKey}(\lambda, L(\lambda), \mathcal{A}'') = 1\big] \geq 2^{-0.4\cdot\lambda^{C_{\mathsf{MoE.Coll}}}}.$$

Proof. It is straightforward to reduce $\mathcal{G}$ to $\mathsf{Moe-Coll-PuncKey}$, which is proven secure in the proof of Theorem 24. See also Claim 16. □

The above constitutes a contradiction by Theorem 24, therefore this completes the security proof.

9 Signature Scheme with Copy-Protected Keys

In this section, we define signature schemes with copy-protected signing keys. Then, we give our construction based on coset states and prove it secure.

9.1 Definitions

Definition 39 (Signature Scheme with Copy-Protected Secret Keys)**.** *A signature scheme with copy-protected secret keys consists of the following efficient algorithms.*

- $\mathsf{KeyGen}(1^\lambda)$*: Takes in the security parameter, output a classical signing key* sk *and a classical verification key* vk.

- $\mathsf{QKeyGen}(sk)$*: Takes as input the classical signing key and outputs a quantum signing key.*

- $\mathsf{Sign}(\mathsf{R}_{\mathsf{sk}}, m)$: *Takes in a quantum signing key and a message m, outputs a classical signature on m.*
- $\mathsf{Ver}(vk, m, sig)$: *Takes in the verification key, a message $m \in \mathcal{M}$ and a claimed signature sig on m, outputs 1 (accept) or 0 (reject).*

We require correctness.

Correctness *For all messages $m \in \mathcal{M}$,*

$$\Pr\left[\mathsf{Ver}(vk, sig) = 1 : \begin{array}{c} sk, vk \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{R}_{\mathsf{sk}} \leftarrow \mathsf{QKeyGen}(sk) \\ sig \leftarrow \mathsf{Sign}(\mathsf{R}_{\mathsf{sk}}, m) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Definition 40 (Pseudodeterministic Signatures)**.** *A signature scheme is said to be* pseudodeterministic *if for any value of sk, vk in the support induced by* KeyGen*, for any message $m \in \mathcal{M}$, there exists a fixed signature $sig_{sk,vk,m}$ such that*

$$\Pr\left[sig = sig_{sk,vk,m} : \begin{array}{c} sk, vk \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{R}_{\mathsf{sk}} \leftarrow \mathsf{QKeyGen}(sk) \\ sig \leftarrow \mathsf{Sign}(\mathsf{R}_{\mathsf{sk}}, m) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

As observed by [LLQZ22], a pseudodeterministic signature scheme, along with Lemma 1, means that we can implement the signing in a way such that the quantum secret key is only negligibly disturbed. Thus, we can reuse the key to sign any polynomial number of times. Our scheme (Section 9.2) will be pseudodeterministic.

We now define anti-piracy security for signature schemes, similar to our PKE definition (Definition 24).

Definition 41 (Anti-Piracy Security for Signature Schemes)**.** *Let* DS *be a signature scheme with copy-protected secret keys. Consider the following game between the challenger and an adversary $\mathcal{A}$.*

$\mathsf{SignatureAntiPiracy}(\lambda, \mathcal{A})$

1. *The challenger runs $sk, vk \leftarrow \mathsf{DS.Setup}(1^\lambda)$ and submits vk to the adversary.*
2. *For multiple rounds, $\mathcal{A}$ makes quantum key queries. For each query, the challenger generates a key as $\mathsf{R} \leftarrow \mathsf{DS.QKeyGen}(sk)$ and submits R to the adversary.*
3. *$\mathcal{A}$ outputs a $(k+1)$-partite register $\mathsf{R}_{\mathsf{adv}}$ and freeloader unitaries $\{U_\ell\}_{\ell \in [k+1]}$ where k is the number of queries it made.*
4. *The challenger executes the following for each $\ell \in [k+1]$.*
   4.1. *$m_\ell \leftarrow \mathcal{M}$.*
   4.2. *$sig_\ell \leftarrow \mathsf{U}_{quantum}(U_\ell, \mathsf{R}_{\mathsf{adv}}[\ell], m_\ell)$.*
   4.3. *Check if $\mathsf{DS.Ver}(vk, m_\ell, sig_\ell) = 1$.*
5. *The challenger outputs 1 if and only if all the checks pass.*

We say that DS *satisfies anti-piracy security if for any QPT adversary $\mathcal{A}$,*

$$\Pr[\mathsf{SignatureAntiPiracy}(\lambda, \mathcal{A}) = 1] \leq \mathsf{negl}(\lambda).$$

9.2 Construction

In this section, we present our construction. Assume the existence of following primitives where we set $\nu(\lambda) = 2^{-6\lambda} \cdot 2^{-8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$.

- F, prefix puncturable extracting PRF (Definition 4) with error $2^{-\lambda-1}$ for min-entropy $s_2(\lambda)+s_3(\lambda)$, with input length $m(\lambda)$ and output length $n(\lambda)$,
- $i\mathcal{O}$, indistinguishability obfuscation scheme that is $\nu(\lambda)$-secure against $2^{5\lambda} \cdot 2^{8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$-time adversaries,
- IBE, identity-based encryption scheme for the identity space $\mathcal{ID} = \{0,1\}^\lambda$ (Definition 15) that is $\nu(\lambda)$-secure against $2^{5\lambda} \cdot 2^{8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$-time adversaries,
- F_1, puncturable PRF family with input length λ and output length same as the size of the randomness used by $\mathsf{CosetGen}$ (Definition 14), that is $\nu(\lambda)$-secure against $2^{5\lambda} \cdot 2^{8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$-time adversaries,
- F_2, puncturable PRF family with input length λ and output length same as the size of the randomness used by $\mathsf{IBE.Enc}$ that is $\nu(\lambda)$-secure against $2^{5\lambda} \cdot 2^{8\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$-time adversaries,[38]
- CCObf, compute-and-compare obfuscation for $2^{-\lambda^{0.2 \cdot C_{\mathsf{MoE.Coll}}}}$-unpredictable distributions that is $2^{-2\lambda-1} \cdot 2^{-2\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$-secure against $2^{3\lambda} \cdot 2^{2\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$-time adversaries,
- F_3, puncturable statistically injective PRF with error probability $2^{-\lambda}$ with input length $s_3(\lambda)$ and output length $s_2(\lambda)$,
- F_4, puncturable PRF with input length $s_2(\lambda)$ and output length $s_3(\lambda)$,
- G_1, a pseudorandom generator with input length $n(\lambda)$ and output length $n(\lambda)$ plus the key size of the PRF F_2,
- G_2, a pseudorandom generator with input length $s_1(\lambda)/2$ and output length $s_1(\lambda)$,
- G_3, a pseudorandom generator with input length λ and output length $2 \cdot \lambda$,
- f, a subexponentially secure injective one-way function with input space $\{0,1\}^{n(\lambda)}$.

We also set the parameters from above as follows:

- $n(\lambda) = \lambda$,
- $s_1(\lambda) = c_L(\lambda)$,
- $s_3(\lambda) - s_1(\lambda) - 2\lambda$ to be larger than the size of the obfuscations (of the program Q) defined in Definition 42,
- $s_2(\lambda) \geq 2 \cdot s_3(\lambda) + \lambda$,
- $s_2(\lambda) + s_3(\lambda) \geq n(\lambda) + 2\lambda + 4$,
- $m(\lambda) = s_1(\lambda) + s_2(\lambda) + s_3(\lambda)$.

[38] We also assume that F_2 has uniformly random keys (when not punctured), that is, the key generation algorithm $F_2.\mathsf{KeyGen}$ simply samples and outputs a uniformly random string. This is satisfied by the puncturable PRF constructions based on one-way functions we are using.

As in our other schemes, while some of our security assumptions above are exponential with specific exponents, all of these assumptions can be based solely on subexponential hardness for any exponent, since we can always scale the security parameter by a polynomial factor when instantiating the underlying primitives.

Set $L(\lambda) = \lambda$ and therefore $c_L(\lambda) = 24 \cdot \lambda^3$ (see Theorem 24). We also assume that all obfuscated programs in the construction and in the proof are appropriately padded.

We now give our signature scheme with copy-protected signing keys, for the message space $\mathcal{M} = \{0,1\}^{m(\lambda)}$.

$\underline{\mathsf{DS.Setup}(1^\lambda)}$

1. Sample PRF keys $K \leftarrow F.\mathsf{KeyGen}(1^\lambda)$ and $K_i \leftarrow F_i.\mathsf{KeyGen}(1^\lambda)$ for $i \in \{1,3,4\}$.
2. Sample $cpk, csmk \leftarrow \mathsf{IBE.Setup}(1^\lambda)$.
3. Sample $\mathsf{OPVer} \leftarrow i\mathcal{O}(\mathsf{PVer})$ where PVer is the following program.

```
$\underline{\mathsf{PVer}(m, sig)}$
Hardcoded: $K, K_3, K_4$
      Hidden Trigger Check
  1. Parse $m_1||m_2||m_3 = m$ with $|m_i| = s_i$.
  2. Compute $m_1'||\mathsf{OQ}'||r' = F_4(K_4, m_2) \oplus m_3$.
  3. Check if $m_1' = m_1$ and $m_2 = F_3(K_3, m_1'||\mathsf{OQ}'||r')$. If so, treat $\mathsf{OQ}'$ as a classical
     circuit, output $\mathsf{OQ}'(\mathsf{mode} = \mathsf{verify}, sig||0^{c_L(\lambda)\cdot\lambda})$ and terminate.
      Normal Mode
  4. Parse $y||K_2' = G_1(F(K, m))$ with $|y| = n(\lambda)$.
  5. Output 1 if $f(sig) = f(y)$. Otherwise, output 0.
```

4. Sample $\mathsf{OPMem} \leftarrow i\mathcal{O}(\mathsf{PMem}_{K_1})$, where PMem_{K_1} is the following program.

```
$\underline{\mathsf{PMem}_{K_1}(id, u_1, \dots, u_{c_L(\lambda)}, x)}$
Hardcoded: $K_1$
  1. $(A_i, s_i, s_i')_{i \in [c_L(\lambda)]} \leftarrow \mathsf{CosetGen}(1^{L(\lambda)+\lambda}; F_1(K_1, id))$.
  2. For each $i \in [c_L(\lambda)]$, check if $u_i \in A_i + s_i$ if $(x)_i = 0$ and check if $u_i \in A_i^\perp + s_i'$ if
     $(x)_i = 1$. If any of the checks fail, output 0 and terminate.
  3. Output 1.
```

5. Sample $\mathsf{OPEval} \leftarrow i\mathcal{O}(\mathsf{PEval})$, where PEval is the following program.[39]

```
$\underline{\mathsf{PEval}(m, id, u_1, \dots, u_{c_L(\lambda)})}$
Hardcoded: $\mathsf{OPMem}, cpk, K, K_3, K_4$
      Hidden Trigger Check
```

[39] Note that it is also possible to put the coset generation PRF key K_1 directly inside OPEval due to the $i\mathcal{O}$ security. However, we elect to use OPMem to preserve the similarities to our PKE construction.

1. Parse $m_1||m_2||m_3 = m$ with $|m_i| = s_i$.
2. Compute $m_1'||\mathsf{OQ}'||r' = F_4(K_4, m_2) \oplus m_3$.
3. Check if $m_1' = m_1$ and $m_2 = F_3(K_3, m_1'||\mathsf{OQ}'||r')$. If so, treat OQ' as a classical circuit, output $\mathsf{OQ}'(\mathsf{mode} = \mathsf{eval}, id, u_1, \ldots, u_{c_L(\lambda)})$ and terminate.
Normal Mode
4. Run $\mathsf{OPMem}(id, u_1, \ldots, u_{c_L(\lambda)}, m_1)$. If it outputs 0, output $\perp$ and terminate.
5. Parse $y||K_2' = G_1(F(K, m))$ with $|y| = n(\lambda)$.
6. Output $\mathsf{IBE.Enc}(cpk, id, y; F_2(K_2', id))$.

6. Set $vk = \mathsf{OPVer}$ and $sk = (cmsk, cpk, K_1, \mathsf{OPEval})$.

7. Output (vk, sk).

$\underline{\mathsf{DS.QKeyGen}(sk)}$

1. Parse $(cmsk, cpk, K_1, \mathsf{OPEval}) = sk$.

2. Sample $id \leftarrow \{0,1\}^\lambda$.

3. $(A_i, s_i, s_i')_{i \in [c_L(\lambda)]} = \mathsf{CosetGen}(1^{L(\lambda)+\lambda}; F_1(K_1, id))$.

4. $ck \leftarrow \mathsf{IBE.KeyGen}(cmsk, id)$.

5. Output $\left(\left|A_{i,s_i,s_i'}\right\rangle\right)_{i \in [c_L(\lambda)]}, ck, id, \mathsf{OPEval}$.

$\underline{\mathsf{DS.Sign}(\mathsf{R_{key}}, m)}$

1. Parse $((\mathsf{R}_i)_{i \in [c_L(\lambda)]}, ck, id, \mathsf{OPEval}) = \mathsf{R_{key}}$.

2. Parse $m_1||m_2||m_3 = m$ with $|m_i| = s_i$.

3. For indices $i \in [c_L(\lambda)]$ such that $(m_0)_i = 1$, apply $H^{\otimes \kappa(L(\lambda)+\lambda)}$ to R_i.

4. Run the program OPEval coherently on m, id and $(\mathsf{R}_i)_{i \in [c_L(\lambda)]}$.

5. Measure the output register and denote the outcome by cct.

6. Output $\mathsf{IBE.Dec}(ck, cct)$.

$\underline{\mathsf{DS.Ver}(vk, m, sig)}$

1. Parse $\mathsf{OPVer} = vk$.

2. Output $\mathsf{OPVer}(m, sig)$.

We claim that the construction is correct and secure.

Theorem 36. DS *satisfies correctness (Definition 39) and psuedodeterminism (Definition 40), and hence reusability.*

Theorem 37. DS *satisfies selective*[40] *message existential unforgeability security.*

Theorem 38. DS *satisfies anti-piracy security (Definition 41).*

When we instantiate the assumed building blocks with known constructions, we get the following corollary.

Corollary 8. *Assuming subexponentially secure $i\mathcal{O}$ and subexponentially secure LWE, there exists a signature scheme that satisfies anti-piracy security against unbounded collusion.*

9.3 Proof of Correctness and Reusability

It is easy to see that the scheme has psuedodeterministic signatures. If a message does not satisfy the hidden trigger condition, the signing procedure will output the first $n(\lambda)$ bits of $G_1(F(K, m))$ by the correctness of the $i\mathcal{O}$ and IBE schemes. In particular, we are assuming perfect correctness for the IBE scheme, which is indeed true for our instantiation (Corollary 4). As a result, the IBE ciphertext $\mathsf{IBE.Enc}(cpk, id, y; F_2(K_2', id))$ output by OPEval will decrypt to y, the first $n(\lambda)$ bits of $G_1(F(K, m))$, for any m. If the message m does satisfy the hidden trigger condition, then the output of OPEval is deterministically determined by m (once sk, vk are fixed). Then, we run $\mathsf{IBE.Dec}$ which we can assume to be deterministic since it has perfect correctness. Therefore, psuedodeterministic signatures property indeed holds for all $m \in \mathcal{M}$.

Now, we move onto correctness. The following discussion follows closely to the proof of correctness given by [LLQZ22] for their scheme. First, it is easy to see that the correctness holds for any m that does not satisfy the hidden trigger condition. We will show that, for any fixed message $m \in \mathcal{M}$, with overwhelming probability over the randomness of the scheme (including setup), the message m will not satisfy the hidden trigger condition, thus proving correctness (Definition 39).

Now, fix a message $m \in \mathcal{M}$. Let $\hat{F}_4$ denote the truncated version of F_4 where we only keep the first s_1 bits, which is also a PRF. Similarly, let $\hat{m_3}$ denote the first s_1 bits of m_3. Observe that if $m \in \mathcal{M}$ does not satisfy correctness, then by above it satisfies the hidden trigger condition, and therefore satisfies $\hat{F}_4(K_4, m_2) = m_1 \oplus \hat{m_3}$ with probability $1/\mathsf{poly}(\lambda)$. Therefore, any sequence of messages that does not satisfy correctness gives us a (non-uniform) adversary for the PRF $\hat{F}_4$ where we can simply distinguish an output $\hat{F}_4(K_4, m_2)$ of the PRF from a random value by checking if it is equal to $m_1 \oplus \hat{m_3}$, which would be satisfied by a random value only with exponentially small probability. This breaks the PRF $\hat{F}_4$ with probability $1/\mathsf{poly}(\lambda)$, which is a contradiction.

9.4 Proof of Existential Unforgeability

We prove the security through a series of hybrids, which are binary random variables denoting the outcome of the forgery game and each one is constructed by modifying the previous one.

$\underline{\mathsf{Hyb}_0}$**:** The original selective message existential unforgeability security game.

$\underline{\mathsf{Hyb}_1}$**:** We define the Hyb_1 so that, after the adversary outputs the challenge message m^*, the challenger checks if m^* satisfies the hidden trigger condition, and it terminates if so. As argued in Section 9.3, this can only happen with negligible probability. Hence, $\mathsf{Hyb}_0 \approx \mathsf{Hyb}_1$.

[40] It can also be made by adaptively secure by complexity leveraging and slightly changing the parameters of the underlying primitives, since we are already assuming subexponentially secure primitives.

$\underline{\mathsf{Hyb}_2}$**:** We first compute $g^* = G_1(F(K, m^*))$, and parse $y^*||K_2^* = g^*$ with $|y^*| = n(\lambda)$ and we set $z^* = f(y^*)$. Finally, we now sample OPVer as $\mathsf{OPVer} \leftarrow i\mathcal{O}(\mathsf{PVer}')$

```
$\underline{\mathsf{PVer}'(m, sig)}$
  Hardcoded: $K\{m^*\}, K_3, K_4, m^*, z^*$
    Hidden Trigger Check
  1. Parse $m_1||m_2||m_3 = m$ with $|m_i| = s_i$.
  2. Compute $m_1'||\mathsf{OQ}'||r' = F_4(K_4, m_2) \oplus m_3$.
  3. Check if $m_1' = m_1$ and $m_2 = F_3(K_3, m_1'||\mathsf{OQ}'||r')$. If so, treat $\mathsf{OQ}'$ as a classical circuit,
     output $\mathsf{OQ}'(\mathsf{mode} = \mathsf{verify}, sig||0^{c_L(\lambda)\cdot\lambda})$ and terminate.
    Normal Mode
  4. If $m = m^*$: Output 1 if $f(sig) = z^*$ and output 0 if $f(sig) \neq z^*$, and terminate.
  5. Parse $y||K_2' = G_1(F(K\{m^*\}, m^*))$ with $|y| = n(\lambda)$.
  6. Output 1 if $f(sig) = f(y)$. Otherwise, output 0.
```

By the punctured key correctness of F, the functionality of PVer did not change. Thus, $\mathsf{Hyb}_1 \approx \mathsf{Hyb}_2$ by the security of $i\mathcal{O}$.

$\underline{\mathsf{Hyb}_3}$**:** We now sample g^* uniformly at random. Observe that PVer only has the punctured key $K\{m^*\}$, and the signing oracle only answers queries for messages $m \neq m^*$, which can also be simulated using $K\{m^*\}$ rather than K. Thus, we have $\mathsf{Hyb}_2 \approx \mathsf{Hyb}_3$ by the punctured key security of F and the security of the PRG G_1.

We claim that $\Pr[\mathsf{Hyb}_3 = 1] \leq \mathsf{negl}(\lambda)$. Observe that, since m^* is not a hidden trigger input, $\mathsf{Hyb}_3 = 1$ occurs if and only if the forged signature sig^* output by the adversary satisfies $f(sig^*) = z^*$, where $z^* = f(y^*)$ and y^* is uniformly at random. Therefore, $\Pr[\mathsf{Hyb}_3 = 1] \leq \mathsf{negl}(\lambda)$ follows by the security of the one-way function f.

9.5 Proof of Anti-Piracy Security

In this section, we prove Theorem 38.

First, we show that hidden trigger inputs are indistinguishable from uniformly random challenge strings, even when the adversary gets a (obfuscated) program that allows it to generate its own hidden trigger inputs.

Definition 42 (Hidden Trigger Inputs)**.** *Let* $\mathsf{GenTrigger}_{K,K_3,K_4,\mathsf{OPMem},cpk}$ *be the following program, where the hardcoded values are as in the signature scheme construction (Section 9.2). The input format to the program will be clear from context.*

```
$\underline{\mathsf{GenTrigger}_{K,K_3,K_4,\mathsf{OPMem},cpk}(r_1, r_2, r_3)}$
  Hardcoded: $K, K_3, K_4, \mathsf{OPMem}, cpk$
  1. Parse $x_1||x_2||x_3 = G_2(r_1)$ with $|x_i| = s_i$.
  2. Parse $y||K_2' = G_1(F(K, x))$ with $|y| = n(\lambda)$.
```

```
3. $\mathsf{OQ} \leftarrow i\mathcal{O}(Q_{cpk,\mathsf{OPMem},x_1,K'_2,y}; r_3)$.
4. $x'_2 = F_3(K_3, x_1||\mathsf{OQ}||G_3(r_2))$.
5. $x'_3 = F_4(K_4, x'_2) \oplus (x_1||\mathsf{OQ}||G_3(r_2))$.
6. Output $x_1||x'_2||x'_3$.
```

The circuit $Q_{cpk,\mathsf{OPMem},x_1,K'_2,y}$ used above is the following. Note that it contains hardcoded values that are computed during the execution of GenTrigger.

```
$Q_{cpk,\mathsf{OPMem},x_1,K'_2,y}(\mathsf{mode}, w)$
  Hardcoded: $cpk, \mathsf{OPMem}, x_1, K'_2, y$
  1. If $\mathsf{mode} = \mathsf{eval}$:
      1. Parse $id, u_1, \ldots, u_{c_L(\lambda)} = w$.
      2. Run $\mathsf{OPMem}(id, u_1, \ldots, u_{c_L(\lambda)}, x_1)$. If it outputs $0$, output $\bot$ and terminate.
      3. Output $\mathsf{IBE.Enc}(cpk, id, y; F_2(K'_2, id))$.
  2. If $\mathsf{mode} = \mathsf{check}$, parse $sig||0^{c_L(\lambda)\cdot\lambda} = w$ and check if $f(sig) = f(y)$. If so, output $1$, and
     otherwise output $0$.
```

Lemma 12. *Let* DS *be the signature scheme from Section 9.2 and let $a(\lambda)$ denote the length of the randomness used by $i\mathcal{O}$ to obfuscate Q in Definition 42. Consider the following experiment, parameterized by $\ell(\lambda)$.*

```
$\mathsf{HiddenTriggerExp}(\lambda, \mathcal{A}, \ell(\lambda), b)$
1. The challenger runs $sk, vk \leftarrow \mathsf{DS.Setup}(1^\lambda)$ and submits $vk$ to the adversary.
2. For multiple rounds, $\mathcal{A}$ makes quantum key queries. For each query, the challenger generates
   a key as $\mathsf{R} \leftarrow \mathsf{DS.QKeyGen}(sk)$ and submits $\mathsf{R}$ to the adversary.
3. The adversary outputs a register $\mathsf{R}_{\mathsf{adv}}$.
4. Sample $\mathsf{OGenTrigger} \leftarrow i\mathcal{O}(\mathsf{GenTrigger})$.
5. For $i = 1$ to $\ell$:
    1. Sample $r_1^i \leftarrow \{0,1\}^{s_1(\lambda)/2}$.
    2. Sample $r_2^i \leftarrow \{0,1\}^{\lambda}$.
    3. Sample $r_3^i \leftarrow \{0,1\}^{a(\lambda)}$.
    4. Set $z^{0,i} = \mathsf{OGenTrigger}(r_1^i, r_2^i, r_3^i)$.
    5. Sample $z^{1,i} \leftarrow \{0,1\}^{m(\lambda)}$.
6. Output $((z^{b,i})_{i\in[\ell]}, \mathsf{OGenTrigger}, \mathsf{R}_{\mathsf{adv}})$.
```

Then, for any polynomial $\ell(\lambda)$,

$$\mathsf{HiddenTriggerExp}(\lambda, \mathcal{A}, \ell(\lambda), 0) \approx_c \mathsf{HiddenTriggerExp}(\lambda, \mathcal{A}, \ell(\lambda), 1).$$

For now, assume Lemma 12 and we prove it at the end of the section. We will prove anti-piracy security through a series of hybrids. Define Hyb_0 to be the original game $\mathsf{SignatureAntiPiracy}(\lambda, \mathcal{A})$.

$\underline{\mathsf{Hyb}_1}$: We now sample m^ℓ for all $\ell \in [k{+}1]$ as hidden triggers (Definition 42): $m^\ell \leftarrow \mathsf{OGenTrigger}(r_1^\ell, r_2^\ell, r_3^\ell)$ where $r_1^\ell, r_2^\ell, r_3^\ell$ are sampled uniformly at random. We get $\mathsf{Hyb}_0 \approx \mathsf{Hyb}_1$ by Lemma 12. Crucially note that at the end of the game in Hyb_1, the forged signatures output by the adversary are verified using OPVer, which is in adversary's view. Hence, the adversary can indeed simulate $\mathsf{Hyb}_0, \mathsf{Hyb}_1$ in the reduction to Lemma 12.

$\underline{\mathsf{Hyb}_2}$: We now sample m^ℓ for $\ell \in [k+1]$ as follows.

1. Sample $r_1^\ell \leftarrow \{0,1\}^{s_1(\lambda)/2}$.
2. Sample $r_2^\ell \leftarrow \{0,1\}^{\lambda}$.
3. Sample $r_3^\ell \leftarrow \{0,1\}^{a(\lambda)}$.
4. Let $x^\ell = G_2(r_1^\ell)$.
5. Parse $x_1^\ell || x_2^\ell || x_3^\ell = x^\ell$ with $|x_i^\ell| = s_i$.
6. Parse $y^\ell || K_2^\ell = G_1(F(K, x^\ell))$ with $|y^\ell| = n(\lambda)$.
7. $\mathsf{OQ}^\ell \leftarrow i\mathcal{O}(Q_{cpk,\mathsf{OPMem},x_1^\ell,K_2^\ell,y^\ell}; r_3^\ell)$.
8. $x_2'^\ell = F_3(K_3, x_1^\ell || \mathsf{OQ}^\ell || G_3(r_2^\ell))$.
9. $x_3'^\ell = F_4(K_4, x_2'^\ell) \oplus (x_1^\ell || \mathsf{OQ}^\ell || G_3(r_2^\ell))$.
10. Set $m^\ell = x_1^\ell || x_2'^\ell || x_3'^\ell$.

Note that this is only a syntactic change, we only unwrapped the sampling of the inputs m^ℓ. Hence, $\mathsf{Hyb}_1 \equiv \mathsf{Hyb}_2$.

$\underline{\mathsf{Hyb}_3}$: We now sample x^ℓ for all $\ell \in [k+1]$ uniformly at random. We get $\mathsf{Hyb}_2 \approx \mathsf{Hyb}_3$ by the security of the PRG G_2.

$\underline{\mathsf{Hyb}_4}$: We now sample all y^ℓ and K_2^ℓ uniformly at random. First, observe that with overwhelming probability, the values x^ℓ for all $\ell \in [k+1]$ will be unique. Then, observe that the inputs x^ℓ to F have each min-entropy $s_2 + s_3$ given the view of the adversary, since only the first part x_1^ℓ is used during the hidden trigger generation and the rest is discarded. Therefore, by our choice of parameters for the extracting PRF F and by the security of PRG G_1, we have $\mathsf{Hyb}_3 \approx \mathsf{Hyb}_4$.

$\underline{\mathsf{Hyb}_5}$: At the end of the game, instead of running OPVer on the forged signatures sig^ℓ output by the adversary, we compare them to y^ℓ. Observe that since m^ℓ are hidden trigger inputs, running OPVer on m^ℓ makes it enter the hidden trigger mode, and it executes OQ^ℓ with $\mathsf{mode} = \mathsf{verify}$, which checks if $f(sig^\ell) = f(y^\ell)$. Since f is injective, this can happen if and only if $y^\ell = sig^\ell$. Hence, $\mathsf{Hyb}_4 \equiv \mathsf{Hyb}_5$.

Finally, observe that the hidden trigger inputs generated above in Hyb_4 are special encodings of $(\mathsf{OQ}^\ell, x_1^\ell)$, which are (*almost*) the same as ciphertexts of our PKE scheme (Section 7.2) encrypting the random messages y^ℓ; and we are comparing the adversary's outputs to y^ℓ. Therefore, the security follows by the random message anti-piracy security (see Section 7.3) of our scheme and we have $\Pr[\mathsf{Hyb}_4 = 1] \leq \mathsf{negl}(\lambda)$.

The only difference between the programs OQ^ℓ here and the ciphertext programs of our PKE is that OQ^ℓ also has a mode that acts as a point on function on the encrypted message: the adversary can query it on some message m' and can check if it equals the encrypted message. See Appendix B for a more detailed discussion on how the security follows from our PKE security proof even the ciphertext programs are modified as such.

9.5.1 Proof of Lemma 12

In this section, we prove Lemma 12. We will only prove the case $\ell = 1$ - the general case for any polynomial $\ell(\lambda)$ follows easily by the hybrid lemma since $\mathsf{OGenTrigger}$ is given to the adversary.

We follow an indirect approach to prove the security by first showing the security of another game, $\mathsf{HiddenTriggerExp}'$. That is, we will first show $\Pr[(\lambda, \mathcal{A}) = 1] \leq 1/2 + \mathsf{negl}(\lambda)$.

$\underline{\mathsf{HiddenTriggerExp}'(\lambda, \mathcal{A})}$

1. The challenger runs $sk, vk \leftarrow \mathsf{DS.Setup}(1^\lambda)$ and submits vk to the adversary.
2. For multiple rounds, $\mathcal{A}$ makes quantum key queries. For each query, the challenger generates a key as $\mathsf{R} \leftarrow \mathsf{DS.QKeyGen}(sk)$ and submits R to the adversary.
3. The adversary outputs a register $\mathsf{R}_{\mathsf{adv}}$.
4. Sample $\mathsf{OGenTrigger} \leftarrow i\mathcal{O}(\mathsf{GenTrigger})$.
5. Sample $r_1^* \leftarrow \{0,1\}^{s_1(\lambda)/2}$.
6. Sample $r_2^* \leftarrow \{0,1\}^{\lambda}$.
7. Sample $r_3^* \leftarrow \{0,1\}^{a(\lambda)}$.
8. Let $x^* = G_2(r_1^*)$.
9. Parse $x_1^*||x_2^*||x_3^* = x^*$ with $|x_i^*| = s_i$.
10. Parse $y^*||K_2^* = G_1(F(K, x^*))$ with $|y^*| = n(\lambda)$.
11. $\mathsf{OQ}^* \leftarrow i\mathcal{O}(Q_{cpk,\mathsf{OPMem},x_1^*,K_2^*,y^*}; r_3^*)$.
12. $\tilde{r} = G_3(r_2^*)$.
13. $x_2'^* = F_3(K_3, x_1^*||\mathsf{OQ}^*||\tilde{r})$.
14. $x_3'^* = F_4(K_4, x_2'^*) \oplus (x_1^*||\mathsf{OQ}^*||\tilde{r})$.
15. Set $z^0 = x_1^*||x_2'^*||x_3'^*$.
16. Set $z^1 = x_1^*||x_2^*||x_3^*$.
17. Sample $b \leftarrow \{0,1\}$.
18. Submit $z^b, \mathsf{OGenTrigger}$ to the adversary $\mathcal{A}$ and the adversary outputs a bit b'.
19. Output 1 if and only if $b' = b$.

First, note that the security of $\mathsf{HiddenTriggerExp}'$ implies indistinguishability between z^0 and z^1 (given the adversary's state and $\mathsf{OGenTrigger}$) by the usual elementary argument. Since r_1^* is random and G_2 is a secure PRG, we also obtain indistinguishability between z^1 and a random x. Combining these two gives us Lemma 12 (for $\ell = 1$) as desired.

We now prove security of $\mathsf{HiddenTriggerExp}'$ through a series of hybrids, each of which is constructed by modifying the previous one. In the hybrids below, whenever we have a check in an obfuscated program where a variable is compared to multiple different values, or a PRF key is punctured at various values, we assume that these are (implicitly) coded in lexicographical order of these values[41] to have symmetry which will be needed in the last hybrid.

$\underline{\mathsf{Hyb}_0}$**:** $\mathsf{HiddenTriggerExp}'(\lambda, \mathcal{A})$.

$\underline{\mathsf{Hyb}_1}$**:** We now sample x^* uniformly at random from $\{0,1\}^{m(\lambda)}$ instead of setting $x^* = G_2(r_1^*)$. By the security of G_2, we get $\mathsf{Hyb}_0 \approx \mathsf{Hyb}_1$.

$\underline{\mathsf{Hyb}_2}$**:** We now sample $\mathsf{OGenTrigger}$ as $\mathsf{OGenTrigger} \leftarrow i\mathcal{O}(\mathsf{GenTrigger}')$ where $K\{x_1^*||\cdot\}$ is the prefix punctured key sampled as $K\{x_1^*||\cdot\} \leftarrow F.\mathsf{Puncture}(K, x_1^*)$.

```
$\underline{\mathsf{GenTrigger}'_{K\{x_1^*||\cdot\},K_3,K_4,\mathsf{OPMem},cpk}(r_1,r_2,r_3)}$
    **Hardcoded:** $K\{x_1^*||\cdot\}, K_3, K_4, \mathsf{OPMem}, cpk$

    1. Parse $x_1||x_2||x_3 = G_2(r_1)$ with $|x_i| = s_i$.
    2. Parse $y||K_2' = F(K\{x_1^*||\cdot\}, x)$ with $|y| = n(\lambda)$.
    3. $\mathsf{OQ} \leftarrow i\mathcal{O}(Q_{cpk,\mathsf{OPMem},x_1,K_2',y}; r_3)$.
    4. $x_2' = F_3(K_3, x_1||\mathsf{OQ}||G_3(r_2))$.
    5. $x_3' = F_4(K_4, x_2') \oplus (x_1||\mathsf{OQ}||G_3(r_2))$.
    6. Output $x_1||x_2'||x_3'$.
```

We claim $\mathsf{Hyb}_0 \approx \mathsf{Hyb}_1$. Observe that, by the prefix punctured key correctness of F, the functionality of $\mathsf{GenTrigger}$ can only possible change if the input x is such that the first $s_1(\lambda)$ bits of $G_2(r_1)$ equals x_1^*. However, the image set of G_2 has size at most $2^{s_1(\lambda)/2}$, and hence the same is true for the truncated version where we only keep the first $s_1(\lambda)$ bits of the outputs. However, x^1* is sampled uniformly at random from $\{0,1\}^{s_1(\lambda)}$. Hence, the probability that the condition above occurs is at most $2^{-s_1(\lambda)/2}$ where $s_1(\lambda)$ is polynomial in λ. Hence, with overwhelming probability, the functionality of $\mathsf{GenTrigger}$ does not change, and $\mathsf{Hyb}_1 \approx \mathsf{Hyb}_2$ follows by the security of $i\mathcal{O}$.

$\underline{\mathsf{Hyb}_3}$**:** We now sample OPVer as $\mathsf{OPVer} \leftarrow i\mathcal{O}(\mathsf{PVer}')$ and OPEval as $\mathsf{OPEval} \leftarrow i\mathcal{O}(\mathsf{PEval}')$.

```
$\underline{\mathsf{PVer}'(m, sig)}$
    **Hardcoded:** $K\{z^0, z^1\}, K_3, K_4, K_5$

        **Hidden Trigger Check**
```

[41] To re-emphasize, we order them according to the hardcoded value, not the variable name that denotes the hardcoded value.

```
    1. If $m = z^0$ or $m = z^1$, output $\mathsf{OQ}^*(\mathsf{mode} = \mathsf{verify}, sig||0^{c_L(\lambda)\cdot\lambda})$ and terminate.
    2. Parse $m_1||m_2||m_3 = m$ with $|m_i| = s_i$.
    3. Compute $m'_1||\mathsf{OQ}'||r' = F_4(K_4, m_2) \oplus m_3$.
    4. Check if $m'_1 = m_1$ and $m_2 = F_3(K_3, m'_1||\mathsf{OQ}'||r')$. If so, treat $\mathsf{OQ}'$ as a classical circuit,
       output $\mathsf{OQ}'(\mathsf{mode} = \mathsf{verify}, sig||0^{c_L(\lambda)\cdot\lambda})$ and terminate.
    Normal Mode
    5. Parse $y||K'_2 = G_1(F(K\{z^0, z^1\}, m))$ with $|y| = n(\lambda)$.
    6. Output 1 if $f(sig) = f(y)$. Otherwise, output 0.
```

```
$\mathsf{PEval}'(m, id, u_1, \ldots, u_{c_L(\lambda)})$
  Hardcoded: $\mathsf{OPMem}, cpk, K\{z^0, z^1\}, K_3, K_4, K_5$
    Hidden Trigger Check
    1. If $m = z^0$ or $m = z^1$, output $\mathsf{OQ}^*(\mathsf{mode} = \mathsf{eval}, id, u_1, \ldots, u_{c_L(\lambda)})$ and terminate.
    2. Parse $m_1||m_2||m_3 = m$ with $|m_i| = s_i$.
    3. Compute $m'_1||\mathsf{OQ}'||r' = F_4(K_4, m_2) \oplus m_3$.
    4. Check if $m'_1 = m_1$ and $m_2 = F_3(K_3, m'_1||\mathsf{OQ}'||r')$. If so, treat $\mathsf{OQ}'$ as a classical circuit,
       output $\mathsf{OQ}'(\mathsf{mode} = \mathsf{eval}, id, u_1, \ldots, u_{c_L(\lambda)})$ and terminate.
    Normal Mode
    5. Run $\mathsf{OPMem}(id, u_1, \ldots, u_{c_L(\lambda)}, m_1)$. If it outputs 0, output $\perp$ and terminate.
    6. Parse $y||K'_2 = G_1(F(K\{z^0, z^1\}, m))$ with $|y| = n(\lambda)$.
    7. Output $\mathsf{IBE.Enc}(cpk, id, y; F_2(K'_2, id))$.
```

By the punctured key correctness of F, the functionality of PVer can only possibly change if $m = z^0$ or $m = z^1$. If $m = z^0$, then the original program would execute the hidden trigger mode and hence would execute OQ^* due to the way we prepared z^0, hence the functionality would be the same. If $m = z^1$, then the original program would execute the normal mode, since with overwhelming probability a random input would not satisfy the hidden trigger condition[42]. In that case, the original program's output would again be the same as the output of OQ^* due to the way OQ^* is prepared. Hence, $\mathsf{Hyb}_2 \approx \mathsf{Hyb}_3$ follows by the security of $i\mathcal{O}$.

$\underline{\mathsf{Hyb}_4}$: We now sample y^* and K_2^* uniformly at random instead of computing them as $y^*||K_2^* = G_1(F(K, x^*))$. Since the adversary only has the punctured keys $K\{z^0, z^1\}$ and $K\{x_1^*||\cdot\}$ where $z^1 = x^*$ and x_1^* is a prefix of x^*, we have that $F(K, x^*)$ is pseudorandom given the adversary's view, by the punctured key security of F. Then, we invoke the security of G_1 and conclude $\mathsf{Hyb}_3 \approx \mathsf{Hyb}_4$.

[42] As discussed in Section 9.3, any sequence of inputs that satisfy the hidden trigger condition with non-negligible probability gives us a way of breaking the security of PRF F_4. Since an adversary can easily sample random inputs, we could break the security of F_4 if random inputs satisfied the hidden trigger condition with non-negligible probaiblity.

$\underline{\mathsf{Hyb}_5}$: We now sample OPVer as $\mathsf{OPVer} \leftarrow i\mathcal{O}(\mathsf{PVer}'')$ and OPEval as $\mathsf{OPEval} \leftarrow i\mathcal{O}(\mathsf{PEval}'')$.

```
$\underline{\mathsf{PVer}''(m, sig)}$
    **Hardcoded:** $K\{z^0, z^1\}, K_3, K_4\{x_2^*, x_2'^*\}, K_5$
        **Hidden Trigger Check** If $m = z^0$ or $m = z^1$, output $\mathsf{OQ}^*(\mathsf{mode} = \mathsf{verify}, sig||0^{c_L(\lambda)\cdot\lambda})$ and terminate.
    1. Parse $m_1||m_2||m_3 = m$ with $|m_i| = s_i$.
    2. If $m_2 = x_2'^*$ or $m_2 = x_2^*$, jump to Normal Mode.
    3. Compute $m_1'||\mathsf{OQ}'||r' = F_4(K_4\{x_2^*, x_2'^*\}, m_2) \oplus m_3$.
    4. Check if $m_1' = m_1$ and $m_2 = F_3(K_3, m_1'||\mathsf{OQ}'||r')$. If so, treat $\mathsf{OQ}'$ as a classical circuit,
       output $\mathsf{OQ}'(\mathsf{mode} = \mathsf{verify}, sig||0^{c_L(\lambda)\cdot\lambda})$ and terminate.
       **Normal Mode**
    5. Parse $y||K_2' = G_1(F(K\{z^0, z^1\}, m))$ with $|y| = n(\lambda)$.
    6. Output 1 if $f(sig) = f(y)$. Otherwise, output 0.
```

```
$\underline{\mathsf{PEval}''(m, id, u_1, \ldots, u_{c_L(\lambda)})}$
    **Hardcoded:** $\mathsf{OPMem}, cpk, K\{z^0, z^1\}, K_3, K_4\{x_2^*, x_2'^*\}, K_5$
        **Hidden Trigger Check**
    1. If $m = z^0$ or $m = z^1$, output $\mathsf{OQ}^*(\mathsf{mode} = \mathsf{eval}, id, u_1, \ldots, u_{c_L(\lambda)})$ and terminate.
    2. Parse $m_1||m_2||m_3 = m$ with $|m_i| = s_i$.
    3. If $m_2 = x_2'^*$ or $m_2 = x_2^*$, jump to Normal Mode.
    4. Compute $m_1'||\mathsf{OQ}'||r' = F_4(K_4\{x_2^*, x_2'^*\}, m_2) \oplus m_3$.
    5. Check if $m_1' = m_1$ and $m_2 = F_3(K_3, m_1'||\mathsf{OQ}'||r')$. If so, treat $\mathsf{OQ}'$ as a classical circuit,
       output $\mathsf{OQ}'(\mathsf{mode} = \mathsf{eval}, id, u_1, \ldots, u_{c_L(\lambda)})$ and terminate.
       **Normal Mode**
    6. Run $\mathsf{OPMem}(id, u_1, \ldots, u_{c_L(\lambda)}, m_1)$. If it outputs 0, output $\perp$ and terminate.
    7. Parse $y||K_2' = G_1(F(K\{z^0, z^1\}, m))$ with $|y| = n(\lambda)$.
    8. Output $\mathsf{IBE.Enc}(cpk, id, y; F_2(K_2', id))$.
```

We will first consider the modified versions of PEval'' and PVer'' where the PRF key K_4 is not punctured at $x_2^*, x_2'^*$ and argue that these versions have the same functionality as PEval' and PVer'. Then, it is easy to see that puncturing K_4 at $x_2^*, x_2'^*$ does not change their functionalities by the punctured key correctness of F_4.

We argue that the newly added skip conditions $m_2 = x_2'^*$ or $m_2 = x_2^*$ does not change the functionalities of $\mathsf{PVer}, \mathsf{PEval}$ except with negligible probability. First, let us consider the check $m_2 = x_2^*$. This new check can only possibly change the functionalities of the programs if we also have $m_2 = F_3(K_3, m_1'||\mathsf{OQ}'||r')$ along with $m_2 = x_2^*$, since then the new program jumps to the normal mode while the old program would possibly execute the hidden trigger mode. However, note that x_2^* is sampled independently uniformly at random from $\{0,1\}^{s_2(\lambda)}$, whereas for any fixing of K_3, the image set of $F_3(K_3, \cdot)$ has size $2^{s_3(\lambda)} \leq 2^{(s_2(\lambda)-\lambda)/2}$. Hence, even the probability that x_2^* is in the image of $F_3(K_3, \cdot)$ is at most $2^{-(s_2(\lambda)-\lambda)/2}$.

Now, let us consider the check $m_2 = x_2^{\prime *}$. As above, this new check can only possibly change the functionalities of the programs if we also have $m_2 = F_3(K_3, m_1'||\mathsf{OQ}'||r')$ along with $m_2 = x_2^{\prime *}$ and $m_1 = m_1'$ where $m_1'||\mathsf{OQ}'||r' = F_4(K_4, m_2) \oplus m_3$. This implies $F_3(K_3, x_1^*||\mathsf{OQ}^*||\tilde{r}) = F_3(K_3, F_4(K_4, m_2) \oplus m_3)$. Assume $F_3(K_3, \cdot)$ is an injective function, which is indeed true with probability $1 - 2^{-\lambda}$ since F_3 is a statistically injective PRF. Then, we get

$$m_3 = (x_1^*||\mathsf{OQ}^*||\tilde{r}) \oplus F_4(K_4, m_2) = (x_1^*||\mathsf{OQ}^*||\tilde{r}) \oplus F_4(K_4, x_2^{\prime *}) = x_3^{\prime *}.$$

Further, $m_1'||\mathsf{OQ}'||r' = F_4(K_4, m_2) \oplus m_3$ along with $m_3 = x_3^{\prime *}$ and $x_2^{\prime *} = m_2$ implies $m_1' = m_1 = x_1^*$. In summary, we get $m_1 = x_1^*, m_2 = x_2^{\prime *}, m_3 = x^{\prime *}$, meaning that $m = z^0$. However, at the beginning of the program we check if $m = z^0$ and jump to normal mode if so. Hence, if $m = z^0$, the program would not even come to this newly added check $m_2 = x_2^{\prime *}$.

By above, we get that except with negligible probability, the functionalities of the obfuscated programs did not change. Thus, $\mathsf{Hyb}_4 \approx \mathsf{Hyb}_5$ by the security of $i\mathcal{O}$.

$\underline{\mathsf{Hyb}_6}$**:** We now sample $\tilde{r}$ uniformly at random from $\{0,1\}^{2\lambda}$. By the security of G_3, we get $\mathsf{Hyb}_5 \approx \mathsf{Hyb}_6$.

$\underline{\mathsf{Hyb}_7}$**:** We now sample $\mathsf{OGenTrigger}$ as $\mathsf{OGenTrigger} \leftarrow i\mathcal{O}(\mathsf{GenTrigger}'')$.

```
$\mathsf{GenTrigger}''_{K\{x_1^*||\cdot\},K_3,K_4\{x_2^{\prime *}\},\mathsf{OPMem},cpk}(r_1, r_2, r_3)$
    Hardcoded: $K\{x_1^*||\cdot\}, K_3, K_4\{x_2^*, x_2^{\prime *}\}, \mathsf{OPMem}, cpk$
      1. Parse $x_1||x_2||x_3 = G_2(r_1)$ with $|x_i| = s_i$.
      2. Parse $y||K_2' = F(K\{x_1^*||\cdot\}, x)$ with $|y| = n(\lambda)$.
      3. $\mathsf{OQ} \leftarrow i\mathcal{O}(Q_{cpk,\mathsf{OPMem},x_1,K_2',y}; r_3)$.
      4. $x_2' = F_3(K_3, x_1||\mathsf{OQ}||G_3(r_2))$.
      5. $x_3' = F_4(K_4\{x_2^*, x_2^{\prime *}\}, x_2') \oplus (x_1||\mathsf{OQ}||G_3(r_2))$.
      6. Output $x_1||x_2'||x_3'$.
```

By the punctured key correctness of F_4, the functionality can only possibly change if $x_2' = x_2^{\prime *}$ or if $x_2' = x_2^*$. Assume that $F_3(K_3, \cdot)$ is an injective function, which is indeed true except with probability $2^{-\lambda}$ since F_3 is a statistically injective PRF. Then, $x_2' = x_2^{\prime *}$ implies $(x_1^*||\mathsf{OQ}^*||\tilde{r}) = x_1||\mathsf{OQ}||G_3(r_2)$, and in particular, $\tilde{r} = G_3(r_2)$. However, observe that the image set of G_3 has size at most 2^λ, whereas $\tilde{r}$ is sampled uniformly at random from $\{0,1\}^{2\lambda}$. Thus, with overwhelming probability, $\tilde{r}$ will be outside the image set of G_3, and hence we will not have $x_2' = x_2^{\prime *}$ for any input to $\mathsf{GenTrigger}$. For the case of $x_2' = x_2^*$, observe that x_2^* is independently sampled uniformly at random and is not used anywhere else. Hence, $x_2' = x_2^*$ can only occur with exponentially small probability. Thus, $\mathsf{Hyb}_6 \approx \mathsf{Hyb}_7$ by the security of $i\mathcal{O}$.

$\underline{\mathsf{Hyb}_8}$: We now sample $\mathsf{OGenTrigger}$ as $\mathsf{OGenTrigger} \leftarrow i\mathcal{O}(\mathsf{GenTrigger}''')$.

```
$\underline{\mathsf{GenTrigger}'''_{K\{x_1^*||\cdot\},K_3\{x_1^*||\mathsf{OQ}^*||\tilde{r}\},K_4\{x_2'^*\},\mathsf{OPMem},cpk}(r_1, r_2, r_3)}$
    Hardcoded: $K\{x_1^*||\cdot\}, K_3\{x_1^*||\mathsf{OQ}^*||\tilde{r}\}, K_4\{x_2^*, x_2'^*\}, \mathsf{OPMem}, cpk$
  1. Parse $x_1||x_2||x_3 = G_2(r_1)$ with $|x_i| = s_i$.
  2. Parse $y||K_2' = F(K\{x_1^*||\cdot\}, x)$ with $|y| = n(\lambda)$.
  3. $\mathsf{OQ} \leftarrow i\mathcal{O}(Q_{cpk,\mathsf{OPMem},x_1,K_2',y}; r_3)$.
  4. $x_2' = F_3(K_3\{x_1^*||\mathsf{OQ}^*||\tilde{r}\}, x_1||\mathsf{OQ}||G_3(r_2))$.
  5. $x_3' = F_4(K_4\{x_2^*, x_2'^*\}, x_2') \oplus (x_1||\mathsf{OQ}||G_3(r_2))$.
  6. Output $x_1||x_2'||x_3'$.
```

As above, with overwhelming probability, $\tilde{r}$ will be outside the image set of G_3, and $x_1||\mathsf{OQ}||G_3(r_2) = x_1^*||\mathsf{OQ}^*||\tilde{r}$ will be not be satisfied by any input to $\mathsf{GenTrigger}$. Hence, the functionality of $\mathsf{GenTrigger}$ does not change with overwhelming probability by the punctured key correctness of F_3, and thus $\mathsf{Hyb}_7 \approx \mathsf{Hyb}_8$ follows by the security of $i\mathcal{O}$.

$\underline{\mathsf{Hyb}_9}$: We now sample OPVer as $\mathsf{OPVer} \leftarrow i\mathcal{O}(\mathsf{PVer}''')$ and OPEval as $\mathsf{OPEval} \leftarrow i\mathcal{O}(\mathsf{PEval}''')$.

```
$\underline{\mathsf{PVer}'''(m, sig)}$
    Hardcoded: $K\{z^0, z^1\}, K_3, K_4\{x_2'^*, x_2^*\}, K_5$
      Hidden Trigger Check If $m = z^0$ or $m = z^1$, output $\mathsf{OQ}^*(\mathsf{mode} = \mathsf{verify}, sig||0^{c_L(\lambda)\cdot\lambda})$
      and terminate.
  1. Parse $m_1||m_2||m_3 = m$ with $|m_i| = s_i$.
  2. If $m_2 = x_2'^*$ or $m_2 = x_2^*$, jump to Normal Mode.
  3. Compute $m_1'||\mathsf{OQ}'||r' = F_4(K_4\{x_2^*, x_2'^*\}, m_2) \oplus m_3$.
  4. Check if $m_1'||\mathsf{OQ}'||r' = x_1^*||\mathsf{OQ}^*||\tilde{r}$. If so, jump to Normal Mode.
  5. Check if $m_1' = m_1$ and $m_2 = F_3(K_3, m_1'||\mathsf{OQ}'||r')$. If so, treat $\mathsf{OQ}'$ as a classical circuit,
     output $\mathsf{OQ}'(\mathsf{mode} = \mathsf{verify}, sig||0^{c_L(\lambda)\cdot\lambda})$ and terminate.
      Normal Mode
  6. Parse $y||K_2' = G_1(F(K\{z^0, z^1\}, m))$ with $|y| = n(\lambda)$.
  7. Output 1 if $f(sig) = f(y)$. Otherwise, output 0.
```

```
$\underline{\mathsf{PEval}'''(m, id, u_1, \ldots, u_{c_L(\lambda)})}$
    Hardcoded: $\mathsf{OPMem}, cpk, K\{z^0, z^1\}, K_3, K_4\{x_2'^*, x_2^*\}, K_5$
      Hidden Trigger Check
```

```
1. If $m = z^0$ or $m = z^1$, output $\mathsf{OQ}^*(\mathsf{mode} = \mathsf{eval}, id, u_1, \dots, u_{c_L(\lambda)})$ and terminate.

2. Parse $m_1||m_2||m_3 = m$ with $|m_i| = s_i$.

3. If $m_2 = x_2^{\prime *}$ or $m_2 = x_2^*$, jump to Normal Mode.

4. Compute $m_1'||\mathsf{OQ}'||r' = F_4(K_4\{x_2^*, x_2^{\prime *}\}, m_2) \oplus m_3$.

5. Check if $m_1'||\mathsf{OQ}'||r' = x_1^*||\mathsf{OQ}^*||\tilde{r}$. If so, jump to Normal Mode.

6. Check if $m_1' = m_1$ and $m_2 = F_3(K_3, m_1'||\mathsf{OQ}'||r')$. If so, treat $\mathsf{OQ}'$ as a classical circuit,
   output $\mathsf{OQ}'(\mathsf{mode} = \mathsf{eval}, id, u_1, \dots, u_{c_L(\lambda)})$ and terminate.
   Normal Mode

7. Run $\mathsf{OPMem}(id, u_1, \dots, u_{c_L(\lambda)}, m_1)$. If it outputs 0, output $\perp$ and terminate.

8. Parse $y||K_2' = G_1(F(K\{z^0, z^1\}, m))$ with $|y| = n(\lambda)$.

9. Output $\mathsf{IBE.Enc}(cpk, id, y; F_2(K_2', id))$.
```

We claim that the newly added skip condition does not change the functionality of the programs. First, note that the functionality of these programs can only possibly change if $m_1'||\mathsf{OQ}'||r' = x_1^*||\mathsf{OQ}^*||\tilde{r}$, $m_1' = m_1$ and $m_2 = F_3(K_3, m_1'||\mathsf{OQ}'||r')$, since then the old programs execute the hidden trigger mode whereas the new program jumps to the normal mode. However, if these conditions are satisfied, then so is $m_2 = x_2^{\prime *}$ due to the way we prepare $x_2^{\prime *}$. Observe that if $m_2 = x_2^{\prime *}$, then the programs will not come to the newly added check since at the beginning we check if $m_2 = x_2^{\prime *}$ and jump to normal mode if so. Hence, the functionality of the programs did not change and we have $\mathsf{Hyb}_8 \approx \mathsf{Hyb}_9$ by the security of $i\mathcal{O}$.

$\underline{\mathsf{Hyb}_{10}}$: We now sample OPVer as $\mathsf{OPVer} \leftarrow i\mathcal{O}(\mathsf{PVer}'''')$ and OPEval as $\mathsf{OPEval} \leftarrow i\mathcal{O}(\mathsf{PEval}'''')$.

```
$\underline{\mathsf{PVer}''''(m, sig)}$
  Hardcoded: $K\{z^0, z^1\}, K_3\{x_1^*||\mathsf{OQ}^*||\tilde{r}\}, K_4\{x_2^{\prime *}, x_2^*\}, K_5$
    Hidden Trigger Check If $m = z^0$ or $m = z^1$, output $\mathsf{OQ}^*(\mathsf{mode} = \mathsf{verify}, sig||0^{c_L(\lambda)\cdot\lambda})$
    and terminate.

1. Parse $m_1||m_2||m_3 = m$ with $|m_i| = s_i$.

2. If $m_2 = x_2^{\prime *}$ or $m_2 = x_2^*$, jump to Normal Mode.

3. Compute $m_1'||\mathsf{OQ}'||r' = F_4(K_4\{x_2^*, x_2^{\prime *}\}, m_2) \oplus m_3$.

4. Check if $m_1'||\mathsf{OQ}'||r' = x_1^*||\mathsf{OQ}^*||\tilde{r}$. If so, jump to Normal Mode.

5. Check if $m_1' = m_1$ and $m_2 = F_3(K_3\{x_1^*||\mathsf{OQ}^*||\tilde{r}\}, m_1'||\mathsf{OQ}'||r')$. If so, treat $\mathsf{OQ}'$ as a
   classical circuit, output $\mathsf{OQ}'(\mathsf{mode} = \mathsf{verify}, sig||0^{c_L(\lambda)\cdot\lambda})$ and terminate.
   Normal Mode

6. Parse $y||K_2' = G_1(F(K\{z^0, z^1\}, m))$ with $|y| = n(\lambda)$.

7. Output 1 if $f(sig) = f(y)$. Otherwise, output 0.
```

```
$\mathsf{PEval}''''(m, id, u_1, \ldots, u_{c_L(\lambda)})$
    Hardcoded: $\mathsf{OPMem}, cpk, K\{z^0, z^1\}, K_3\{x_1^* || \mathsf{OQ}^* || \tilde{r}\}, K_4\{x_2'^*, x_2^*\}, K_5$
        Hidden Trigger Check
    1. If $m = z^0$ or $m = z^1$, output $\mathsf{OQ}^*(\mathsf{mode} = \mathsf{eval}, id, u_1, \ldots, u_{c_L(\lambda)})$ and terminate.
    2. Parse $m_1 || m_2 || m_3 = m$ with $|m_i| = s_i$.
    3. If $m_2 = x_2'^*$ or $m_2 = x_2^*$, jump to Normal Mode.
    4. Compute $m_1' || \mathsf{OQ}' || r' = F_4(K_4\{x_2^*, x_2'^*\}, m_2) \oplus m_3$.
    5. Check if $m_1' || \mathsf{OQ}' || r' = x_1^* || \mathsf{OQ}^* || \tilde{r}$. If so, jump to Normal Mode.
    6. Check if $m_1' = m_1$ and $m_2 = F_3(K_3\{x_1^* || \mathsf{OQ}^* || \tilde{r}\}, m_1' || \mathsf{OQ}' || r')$. If so, treat $\mathsf{OQ}'$ as a classical circuit, output $\mathsf{OQ}'(\mathsf{mode} = \mathsf{eval}, id, u_1, \ldots, u_{c_L(\lambda)})$ and terminate.
        Normal Mode
    7. Run $\mathsf{OPMem}(id, u_1, \ldots, u_{c_L(\lambda)}, m_1)$. If it outputs 0, output $\perp$ and terminate.
    8. Parse $y || K_2' = G_1(F(K\{z^0, z^1\}, m))$ with $|y| = n(\lambda)$.
    9. Output $\mathsf{IBE.Enc}(cpk, id, y; F_2(K_2', id))$.
```

The functionality of these programs stay the same by the punctured key correctness of K_3. Thus, $\mathsf{Hyb}_9 \approx \mathsf{Hyb}_{10}$ follows by the security of $i\mathcal{O}$.

$\underline{\mathsf{Hyb}_{11}}$**:** We now sample $x_2'^*$ uniformly at random. $\mathsf{Hyb}_{10} \approx \mathsf{Hyb}_{11}$ follows by the punctured key security of F_3.

$\underline{\mathsf{Hyb}_{12}}$**:** We now sample $x_3'^*$ uniformly at random. $\mathsf{Hyb}_{11} \approx \mathsf{Hyb}_{12}$ follows by the punctured key security of F_4.

Finally, we claim that z^0 and z^1 are symmetric in Hyb_{12}. First note that they are both sampled uniformly at random. Further, any check we have in the obfuscated programs compare variables to both z^0 and z^1, which are coded in lexicographical order to keep the symmetry. Finally, any PRF key that is punctured is either punctured at both z^0, z^1, or both $x_2'^*, x_2^*$, or (prefix) punctured at x_1^*, or punctured at $x_1^* || \mathsf{OQ}^* || \tilde{r}$ where OQ^* depends only on y^*, K_2^* and x_1^*. Crucially note that $y^*, K_2^*, \tilde{r}$ are sampled independently uniformly at random, and x_1^* is the first s_1 bits of both z^0 and z^1. Hence, z^0 and z^1 are indeed symmetric in Hyb_{12}, and thus we have $\Pr[\mathsf{Hyb}_{12} = 1] \leq 1/2$, concluding the proof.

10 Pseudorandom Function Family with Copy-Protected Keys

In this section, we define PRF schemes with copy-protected keys. Then, we give our construction based on coset states and prove it secure.

10.1 Definitions

Definition 43 (PRF Scheme with Copy-Protected Secret Keys)**.** *A PRF scheme with copy-protected secret keys consists of the following efficient algorithms.*

- $\mathsf{KeyGen}(1^\lambda)$: *Takes in the security parameter, output a classical key K.*
- $\mathsf{QKeyGen}(K)$: *Takes as input the classical key and outputs a quantum key.*
- $\mathsf{Eval}(\mathsf{R}_K, m)$: *Takes in a quantum key and an input x, outputs a classical value.*

We require correctness.

Correctness *For all inputs x,*

$$\Pr\left[val = F(K,x) : \begin{array}{c} K \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{R}_K \leftarrow \mathsf{QKeyGen}(K) \\ val \leftarrow \mathsf{Eval}(\mathsf{R}_K, x) \end{array} \right] \geq 1 - \mathsf{negl}(\lambda).$$

As observed by [CLLZ21], correctness, along with Lemma 1, means that we can implement the evaluation in a way such that the quantum key is only negligibly disturbed. Thus, we can reuse the key to evaluate the PRF any polynomial number of times.

Definition 44 (Anti-Piracy Security for PRF Schemes)**.** *Consider the following game between the challenger and an adversary $\mathcal{A}$.*

$\underline{\mathsf{PRFAntiPiracy}(\lambda, \mathcal{A})}$

1. *The challenger runs $K \leftarrow \mathsf{KeyGen}(1^\lambda)$.*
2. *For multiple rounds, $\mathcal{A}$ makes quantum key queries. For each query, the challenger generates a key as $\mathsf{R} \leftarrow \mathsf{QKeyGen}(K)$ and submits R to the adversary.*
3. *$\mathcal{A}$ outputs a $(k+1)$-partite register $\mathsf{R}_{\mathsf{adv}}$ and freeloader unitaries $\{U_\ell\}_{\ell \in [k+1]}$ where k is the number of queries it made.*
4. *The challenger executes the following for each $\ell \in [k+1]$.*
   4.1. *Sample $b_\ell \leftarrow \{0,1\}$.*
   4.2. *$x^\ell \leftarrow \{0,1\}^{m(\lambda)}$.*
   4.3. *Set $ch^{0,\ell} = F(K, x^\ell)$ and sample $ch^{1,\ell} \leftarrow \{0,1\}^{n(\lambda)}$.*
   4.4. *$b'_\ell \leftarrow \mathsf{U}_{quantum}(U_\ell, \mathsf{R}_{\mathsf{adv}}[\ell], x^\ell, ch^{b_\ell,\ell})$.*
   4.5. *Check if $b'_\ell = b_\ell$.*
5. *The challenger outputs 1 if and only if all the checks pass.*

We say that the PRF scheme satisfies anti-piracy security if for any QPT adversary $\mathcal{A}$,

$$\Pr[\mathsf{PRFAntiPiracy}(\lambda, \mathcal{A}) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

10.2 Construction

In this section, we present our construction copy-protecting a particular PRF family F. Our construction is the same as our copy-protected signature construction (Section 9), with the verification key removed. We give it in full for completeness.

Assume the existence of following primitives where we set $\nu(\lambda) = 2^{-6\lambda} \cdot 2^{-8\lambda^{0.3 C_{\mathsf{MoE.Coll}}}}$.

- F, prefix puncturable extracting PRF (Definition 4) with error $2^{-\lambda-1}$ for min-entropy $s_2(\lambda) + s_3(\lambda)$, with input length $m(\lambda)$ and output length $n(\lambda)$,
- $i\mathcal{O}$, indistinguishability obfuscation scheme that is $\nu(\lambda)$-secure against $2^{5\lambda} \cdot 2^{8\lambda^{0.3 C_{\mathsf{MoE.Coll}}}}$-time adversaries,
- IBE, identity-based encryption scheme for the identity space $\mathcal{ID} = \{0,1\}^\lambda$ (Definition 15) that is $\nu(\lambda)$-secure against $2^{5\lambda} \cdot 2^{8\lambda^{0.3 C_{\mathsf{MoE.Coll}}}}$-time adversaries,
- F_1, puncturable PRF family with input length λ and output length same as the size of the randomness used by $\mathsf{CosetGen}$ (Definition 14), that is $\nu(\lambda)$-secure against $2^{5\lambda} \cdot 2^{8\lambda^{0.3 C_{\mathsf{MoE.Coll}}}}$-time adversaries,
- F_2, puncturable PRF family with input length λ and output length same as the size of the randomness used by $\mathsf{IBE.Enc}$ that is $\nu(\lambda)$-secure against $2^{5\lambda} \cdot 2^{8\lambda^{0.3 C_{\mathsf{MoE.Coll}}}}$-time adversaries,[43]
- CCObf, compute-and-compare obfuscation for $2^{-\lambda^{0.2 \cdot C_{\mathsf{MoE.Coll}}}}$-unpredictable distributions that is $2^{-2\lambda-1} \cdot 2^{-2\lambda^{0.3 C_{\mathsf{MoE.Coll}}}}$-secure against $2^{3\lambda} \cdot 2^{2\lambda^{0.3 C_{\mathsf{MoE.Coll}}}}$-time adversaries,
- F_3, puncturable statistically injective PRF with error probability $2^{-\lambda}$ with input length $s_3(\lambda)$ and output length $s_2(\lambda)$,
- F_4, puncturable PRF with input length $s_2(\lambda)$ and output length $s_3(\lambda)$,
- G_1, a pseudorandom generator with input length $n(\lambda)$ and output length $n(\lambda)$ plus the key size of the PRF F_2,
- G_2, a pseudorandom generator with input length $s_1(\lambda)/2$ and output length $s_1(\lambda)$,
- G_3, a pseudorandom generator with input length λ and output length $2 \cdot \lambda$,
- f, a subexponentially secure injective one-way function with input space $\{0,1\}^{n(\lambda)}$.

We also set the parameters from above as follows:

- $n(\lambda) = \lambda$,
- $s_1(\lambda) = c_L(\lambda)$,
- $s_3(\lambda) - s_1(\lambda) - 2\lambda$ to be large enough to contain obfuscations of the program Q defined in Definition 52,
- $s_2(\lambda) \geq 2 \cdot s_3(\lambda) + \lambda$,

[43] We also assume that F_2 has uniformly random keys (when not punctured), that is, the key generation algorithm $F_2.\mathsf{KeyGen}$ simply samples and outputs a uniformly random string. This is satisfied by the puncturable PRF constructions based on one-way functions we are using.

- $s_2(\lambda) + s_3(\lambda) \geq n(\lambda) + 2\lambda + 4$,
- $m(\lambda) = s_1(\lambda) + s_2(\lambda) + s_3(\lambda)$.

While some of our security assumptions above are exponential with specific exponents, these assumptions can be based solely on subexponential hardness for any exponent, since we can always scale the security parameter by a polynomial factor when instantiating the underlying primitives.

Set $L(\lambda) = \lambda$ and therefore $c_L(\lambda) = 24 \cdot \lambda^3$ (see Theorem 24). We also assume that all obfuscated programs in the construction and in the proof are appropriately padded.

We now give our PRF scheme with copy-protected keys, for the input space $\{0,1\}^{m(\lambda)}$.

$\underline{\mathsf{KeyGen}(1^\lambda)}$

1. Sample PRF keys $K_0 \leftarrow F.\mathsf{KeyGen}(1^\lambda)$ and $K_i \leftarrow F_i.\mathsf{KeyGen}(1^\lambda)$ for $i \in \{1,3,4\}$.
2. Sample $cpk, csmk \leftarrow \mathsf{IBE.Setup}(1^\lambda)$.
3. Sample $\mathsf{OPMem} \leftarrow i\mathcal{O}(\mathsf{PMem}_{K_1})$, where PMem_{K_1} is the following program.

```
$\underline{\mathsf{PMem}_{K_1}(id, u_1, \ldots, u_{c_L(\lambda)}, x)}$
Hardcoded: $K_1$
  1. $(A_i, s_i, s'_i)_{i \in [c_L(\lambda)]} \leftarrow \mathsf{CosetGen}(1^{L(\lambda)+\lambda}; F_1(K_1, id))$.
  2. For each $i \in [c_L(\lambda)]$, check if $u_i \in A_i + s_i$ if $(x)_i = 0$ and check if $u_i \in A_i^\perp + s'_i$ if
     $(x)_i = 1$. If any of the checks fail, output 0 and terminate.
  3. Output 1.
```

4. Sample $\mathsf{OPEval} \leftarrow i\mathcal{O}(\mathsf{PEval})$, where PEval is the following program.[44]

```
$\underline{\mathsf{PEval}(x, id, u_1, \ldots, u_{c_L(\lambda)})}$
Hardcoded: $\mathsf{OPMem}, cpk, K_0, K_3, K_4$
    Hidden Trigger Check
  1. Parse $x_1||x_2||x_3 = x$ with $|x_i| = s_i$.
  2. Compute $x'_1||\mathsf{OQ}'||r' = F_4(K_4, x_2) \oplus m_3$.
  3. Check if $x'_1 = x_1$ and $x_2 = F_3(K_3, x'_1||\mathsf{OQ}'||r')$. If so, treat $\mathsf{OQ}'$ as a classical circuit,
     output $\mathsf{OQ}'(id, u_1, \ldots, u_{c_L(\lambda)})$ and terminate.
    Normal Mode
  4. Run $\mathsf{OPMem}(id, u_1, \ldots, u_{c_L(\lambda)}, x_1)$. If it outputs 0, output $\perp$ and terminate.
  5. Parse $y||K'_2 = G_1(F(K_0, x))$ with $|y| = n(\lambda)$.
  6. Output $\mathsf{IBE.Enc}(cpk, id, y; F_2(K'_2, id))$.
```

5. Set $K = (cmsk, cpk, K_1, \mathsf{OPEval})$.
6. Output K.

[44] Note that it is also possible to put the coset generation PRF key K_1 directly inside OPEval due to the $i\mathcal{O}$ security. However, we elect to use OPMem to preserve the similarities to our PKE construction.

$\underline{\mathsf{QKeyGen}(K)}$

1. Parse $(cmsk, cpk, K_1, \mathsf{OPEval}) = K$.
2. Sample $id \leftarrow \{0,1\}^\lambda$.
3. $(A_i, s_i, s'_i)_{i \in [c_L(\lambda)]} = \mathsf{CosetGen}(1^{L(\lambda)+\lambda}; F_1(K_1, id))$.
4. $ck \leftarrow \mathsf{IBE.KeyGen}(cmsk, id)$.
5. Output $\left(\left|A_{i,s_i,s'_i}\right\rangle\right)_{i \in [c_L(\lambda)]}, ck, id, \mathsf{OPEval}$.

$\underline{\mathsf{Eval}(\mathsf{R}_{\mathsf{key}}, x)}$

1. Parse $((\mathsf{R}_i)_{i \in [c_L(\lambda)]}, ck, id, \mathsf{OPEval}) = \mathsf{R}_{\mathsf{key}}$.
2. Parse $x_1||x_2||x_3 = x$ with $|x_i| = s_i$.
3. For indices $i \in [c_L(\lambda)]$ such that $(x_0)_i = 1$, apply $H^{\otimes \kappa(L(\lambda)+\lambda)}$ to R_i.
4. Run the program OPEval coherently on x, id and $(\mathsf{R}_i)_{i \in [c_L(\lambda)]}$.
5. Measure the output register and denote the outcome by cct.
6. Output $\mathsf{IBE.Dec}(ck, cct)$.

We claim that the construction is correct and secure.

Theorem 39. *The PRF scheme satisfies correctness and hence reusability.*

Proof. Since our construction is the same as our signature scheme, follows from Section 9.3. □

Theorem 40. *The PRF scheme satisfies PRF security.*

Proof. Our PRF family is a truncation of $G_1(F(K_0, \cdot))$ where G_1 is a PRG and F is a PRF. Therefore, it is easy to see that the resulting function family also satisfies the PRF security game. □

Theorem 41. *The PRF scheme satisfies anti-piracy security.*

Proof. The proof closely follows the anti-piracy security proof our signature scheme, the major difference being that in the PRF case we have a CPA-style game where the adversary is trying to guess a challenge bit and we require that it wins with probability at most $1/2 + \mathsf{negl}(\lambda)$, whereas in the signature game we required negligible winning probability. See Appendix D for the full proof. □

When we instantiate the assumed building blocks with known constructions, we get the following corollary.

Corollary 9. *Assuming subexponentially secure $i\mathcal{O}$ and subexponentially secure LWE, there exists a PRF scheme that satisfies anti-piracy security against unbounded collusion.*

11 Copy-Protection for All Unlearnable Functionalities

In this section, we first reproduce the generalized copy-protection definitions from [ALL+21], and then we show how to copy-protect any unlearnable functionality with respect to a classical oracle.

11.1 Definitions

We now reproduce the relevant definitions from [ALL+21].

Definition 45 (Testing an Oracle-Aided Quantum Program)**.** *Let $\mathcal{F}$ be a family of functions with input length $m(\lambda)$ and output length $n(\lambda)$. Fix some program f from this family, an oracle-aided unitary U and some value st of a classical state maintained by the challenger (which will be defined later). Let $\mathcal{D}$ be an efficient challenge input distribution (over $\{0,1\}^{m(\lambda)}$), and let $\mathcal{O}$ be a quantumly accessible classical oracle that can depend on st. Consider the following mixture $\mathcal{P}$ of binary* projective measurements*, induced by $\mathcal{D}$ and f, U, st, applied on a state ρ.*

1. *Sample $r \leftarrow \mathcal{R}$.*
2. *Run $x \leftarrow \mathcal{D}^{st}(1^\lambda; r)$.*
3. *Execute U on (ρ, ct), let y' be the output.*
4. *Output 1 if $y' = f(x)$. Otherwise, output* 0*.*

Observe that we can efficiently execute the above measurement[45] *for arbitrary given superpositions of r and b values. Therefore, by Section 4, there exists both exact and approximated projective and threshold implementations for $\mathcal{P}$. We write $\mathsf{PI}^{\mathcal{O}}_{\mathcal{D}}$ and $\mathsf{API}^{\mathcal{O},\varepsilon,\delta}_{\mathcal{D}}$ to denote the projective implementation and approximate projective implementation of $\mathcal{P}$, respectively. Similarly, let $\mathsf{TI}^{\mathcal{O}}_{\mathcal{D},\eta}$ and $\mathsf{ATI}^{\mathcal{O},\varepsilon,\delta}_{\mathcal{D},\eta}$ denote the threshold and efficient approximate threshold implementations of $\mathcal{P}$ for a threshold value η.*

While the fixed values f, U, st are omitted from the notation, they will be clear from the context.

Definition 46 (γ-Quantum Unlearnability [ALL+21])**.** *Let $\mathcal{F}$ be a family of functions with input length $m(\lambda)$, and let $\mathcal{D}$ be an input distribution over $\{0,1\}^{m(\lambda)}$. Consider the following game between the challenger and an adversary $\mathcal{A}$.*

$\mathsf{LearningGame}(\lambda, \gamma(\lambda), \mathcal{A})$

1. *The challenger samples a function f from $\mathcal{F}$.*
2. *$\mathcal{A}$ gets oracle access to f.*
3. *$\mathcal{A}$ outputs a quantum register $\mathsf{R}_{\mathsf{adv}}$ and a unitary U.*
4. *The challenger applies $\mathsf{TI}_{\mathcal{D},\gamma}$ to $\mathsf{R}_{\mathsf{adv}}$, outputs the measurement result.*

We say that $(\mathcal{F}, \mathcal{D})$ is γ-unlearnable if for any QPT polynomial $\mathcal{A}$,

$$\Pr[\mathsf{LearningGame}(\lambda, \gamma(\lambda), \mathcal{A}) = 1] \leq \mathsf{negl}(\lambda).$$

[45] More formally, we are actually talking about the measurement where r, b are fixed

Definition 47 (Quantum Copy-Protection Scheme [ALL+21])**.** *Let $\mathcal{F}$ be a family of functions with input length $m(\lambda)$ and output length $n(\lambda)$. A copy-protection scheme for $\mathcal{F}$ consists of the following efficient algorithms.*

- $\mathsf{Setup}(1^\lambda)$*: Takes as input a security parameter and outputs a classical secret key sk,*
- $\mathsf{QGen}(sk, f)$*: Takes in the secret key and a function $f \in \mathcal{F}$, outputs a copy-protected program as a quantum state.*
- $\mathsf{Eval}(\mathsf{R}_{\mathsf{key}}, x)$*: Takes in a copy-protected program and an input, outputs a value from $\{0,1\}^{n(\lambda)}$.*

We require correctness.

Correctness *For all functions $f \in \mathcal{F}$ and inputs $x \in \{0,1\}^{m(\lambda)}$,*

$$\Pr\left[\mathsf{Eval}(\mathsf{R}_{\mathsf{key}}, x) = f(x) : \begin{array}{c} sk \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{R}_f \leftarrow \mathsf{QGen}(sk, f) \end{array}\right] = 1.$$

Definition 48 (γ-Anti-Piracy Security [ALL+21])**.** *Let $\mathcal{F}$ be a family of functions with input length $m(\lambda)$, and let $\mathcal{D}$ be an input distribution over $\{0,1\}^{m(\lambda)}$. Consider a copy-protection scheme (Definition 47) for $\mathcal{F}$ and the following game between the challenger and an adversary $\mathcal{A}$.*

$\underline{\mathsf{AntiPiracyGame}(\lambda, \gamma(\lambda), \mathcal{A})}$

1. *The challenger samples a copy-protection key $sk \leftarrow \mathsf{Setup}(1^\lambda)$.*
2. *The challenger samples a function f from $\mathcal{F}$.*
3. *For multiple rounds, $\mathcal{A}$ makes quantum key queries. For each query, the challenger generates a key as $\mathsf{R} \leftarrow \mathsf{QGen}(sk, f)$ and submits R to the adversary.*
4. *$\mathcal{A}$ outputs a $(k+1)$-partite quantum register $\mathsf{R}_{\mathsf{adv}}$ and freeloader unitaries $\{U\ell\}_{\ell \in [k+1]}$ where k is the number of key queries it made.*
5. *The challenger applies the test*
$$\bigotimes_{\ell \in [k+1]} \mathsf{TI}_{\mathcal{D},\gamma}$$
to $\mathsf{R}_{\mathsf{adv}}$ and outputs 1 if and only if the measurement result is all 1.

We say that the copy-protection scheme satisfies γ-anti-piracy if for any QPT polynomial $\mathcal{A}$,

$$\Pr[\mathsf{AntiPiracyGame}(\lambda, \gamma(\lambda), \mathcal{A}) = 1] \leq \mathsf{negl}(\lambda).$$

11.2 Construction

In this section, we present our copy-protection construction for a family of functions $\mathcal{F}$ with input length $m(\lambda)$ and output length $n(\lambda)$. Assume the existence of following primitive.

- F_1, PRF family with input length λ and output length same as the size of the randomness used by $\mathsf{CosetGen}$ (Definition 14) that is $2^{-2\lambda}$-secure against QPT adversaries.

While we assume exponential security of the above primitive for specific exponents, this assumption can be based solely on subexponential hardness for any exponent, since we can always scale the security parameter by a polynomial factor when instantiating the underlying primitives.

We now give our construction.

$\underline{\mathsf{Setup}(1^\lambda)}$

1. Sample PRF key $K_1 \leftarrow F_1.\mathsf{KeyGen}(1^\lambda)$.
2. Output K_1.

$\underline{\mathsf{QGen}(sk, f)}$

1. Parse $K_1 = sk$.
2. Generate the oracle $\mathcal{O}_f$.

   > $\underline{\mathcal{O}_f(id, x, (v_i)_{i\in[m(\lambda)]})}$
   >
   > **Hardcoded:** K_1, f
   >
   > 1. $(A_i, s_i, s'_i)_{i\in[m(\lambda)]} \leftarrow \mathsf{CosetGen}(1^\lambda, m(\lambda), \lambda; F_1(K_1, id))$.
   > 2. For each $i \in [m(\lambda)]$, check if $v_i \in A_i + s_i$ if $(x)_i = 0$ and check if $v_i \in A_i^\perp + s'_i$ if $(x)_i = 1$. If any of the checks fail, output $\perp$ and terminate.
   > 3. Output $f(x)$.

3. Sample $id \leftarrow \{0,1\}^\lambda$.
4. $(A_i, s_i, s'_i)_{i\in[m(\lambda)]} \leftarrow \mathsf{CosetGen}(1^\lambda, m(\lambda), \lambda; F_1(K_1, id))$.
5. Output $\left(\left|A_{i,s_i,s'_i}\right\rangle\right)_{i\in[m(\lambda)]}, id, \mathcal{O}$.

$\underline{\mathsf{Eval}(\mathsf{R}_{\mathsf{key}}, x)}$

1. Parse $((\mathsf{R}_i)_{i\in[m(\lambda)]}, id, \mathcal{O}) = \mathsf{R}_{\mathsf{key}}$.
2. For indices $i \in [m(\lambda)]$ such that $(x)_i = 1$, apply H^λ to R_i.
3. Run the oracle $\mathcal{O}$ coherently on id, x and $(\mathsf{R}_i)_{i\in[m(\lambda)]}$.
4. Measure the output register and output the measurement outcome.

Correctness with probability 1 follows in a straightforward manner. We claim that the construction is also secure.

Theorem 42. *For any inverse polynomial γ and any function family and challenge input distribution $(\mathcal{F}, \mathcal{D})$ that is γ-unlearnable , the scheme above satisfies strong γ-anti-piracy.*

Proof. The proof follows in a similar manner to the anti-piracy security proof of our PKE scheme (Section 7.3). See Appendix C for the full proof. □

12 Impossibility of Hyperefficient Shadow Tomography

In this section, as a corollary of results, we rule out existence of hyperefficient shadow tomography.

Definition 49 (Hyperefficient Shadow Tomography [Aar18])**.** *Let E denote a uniform quantum circuit family with classical binary output that takes as input $i \in [M]$ and an n-qubit quantum state ρ.*[46] *Then, a shadow tomography procedure takes as input E and $\rho^{\otimes k}$ where k denotes the number of copies, and outputs a quantum circuit*[47] *C such that* $\Pr[\forall i \in [M]\ |C(i) - \Pr[E(i,\rho) = 1]| < \varepsilon] > 1 - \delta$. *The procedure is said to be* hyperefficient *if the number of copies k and the runtime are both* $\mathsf{poly}(n, \log M, \frac{1}{\varepsilon})$.

[Aar18] shows that shadow tomography can be performed using polynomially many copies of ρ, however, the procedure takes exponential time. They leave it as an open question to give a hyperefficient shadow tomography procedure or rule out its existence. [AK07, Aar09, Kre21] rule it out in oracle models, where the procedure has only black-box query access to the measurement circuit $E(i, \rho)$.

[Aar18] also shows that shadow tomography gives a generic attack on copy-protection schemes, and combined with their own sample-efficient shadow tomography procedure, they show that collusion-resistant copy-protection cannot exist without computational assumptions. Later, [SW22] adapts this attack to the case of unclonable decryptors, i.e., copy-protected secret keys for PKE, to conclude its impossibility without computational assumptions.

By Corollary 6, we obtain the following result.

Corollary 10. *Assuming post-quantum subexponentially secure indistinguishability obfuscation and LWE, there cannot exist a hyperefficient shadow tomography algorithm.*

Proof. We prove the result by showing that shadow tomograpghy breaks PKE with copy-protected keys, which we construct in Corollary 6. Our attack is exactly the same as the one given by [SW22], we merely observe that the attack is efficient when the shadow tomography procedure is efficient. We describe it below for completeness.

Let PKE be the PKE scheme with collusion-resistant copy-protected secret keys given in Section 7.2, for 1-bit messages. Define the measurement circuit E to be $\mathsf{PKE.Dec}(ct, \rho)$, where the measurement E_{ct} outputs 1 if the decryption procedure outputs 1 when ρ is given as the input to the key register. Note that E is uniform. Suppose there exists a hyperefficient shadow tomography algorithm. Then, consider the following adversary for the anti-piracy game for PKE. We obtain k keys where k is the number of copies needed by the shadow tomography procedure, which is $\mathsf{poly}(\lambda)$ by assumption. We perform the procedure with $\varepsilon = 1/8$ and $\delta = \frac{1}{2(k+1)}$ to obtain the estimation circuit C. We pick $0, 1$ as our challenge messages, and output C to all $k+1$ freeloaders. When presented with a challenge ciphertext ct, a freeloader runs $C(ct)$ and outputs 1 if outputs a value $> 3/4$, and outputs 0 otherwise. Note that if ct is an encryption of 1, we will have $\Pr[\mathsf{PKE.Dec}(ct, \rho) = 1] \leq 1 - \mathsf{negl}(\lambda)$, hence $C(ct) > 3/4$ with probability $1 - \delta$, and we will correctly decrypt. By the same argument, all the freeloaders will simultaneously correctly decrypt with probability $> 1/2$. Note that the whole attack is efficient by assumption. This breaks the security of PKE, which is a contradiction. □

Remark 1. *We note that to rule out hyperefficient shadow tomography, unbounded collusion-resistant schemes are needed - bounded collusion-resistance ([LLQZ22]) is not sufficient, for the*

[46] That is, E on input i, ρ measures ρ with respect to a binary measurement, which we can denote E_i

[47] More precisely, classical description of a quantum circuit, since otherwise we can just hardwire the state ρ into the circuit.

following reason. Since the number of copies required by hyperefficient shadow tomography procedure can depend on $\log M = \log 2^{|ct|} = |ct|$*, if the ciphertext size grows with the collusion bound* k*, so does the number of copies needed. Hence, the hyperefficient shadow tomography procedure might need* $k+1$ *or more copies to work, in which case we cannot arrive at a contradiction as we did above.*

We also obtain the following result.

Corollary 11. *There exists a quantumly accessible classical oracle relative to which there does not exist a hyperefficient shadow tomography algorithm.*

Proof. In this setting, the set of measurements $\{E_i\}_i$ is given by a quantumly accessible classical oracle $\mathcal{O}$ such that on input i, the oracle outputs the description of the measurement E_i.

By the same argument as above, our collusion-resistant copy-protection scheme for all unlearnable functionalities given in Section 11 implies the stated result. □

13 Acknowledgements

We thank Jiahui Liu for helpful discussions. Alper Çakan was supported by the following grants of Vipul Goyal: NSF award 1916939, DARPA SIEVE program, a gift from Ripple, a DoE NETL award, a JP Morgan Faculty Fellowship, a PNC center for financial services innovation award, and a Cylab seed funding award.

References

[Aar09] Scott Aaronson. Quantum copy-protection and quantum money. In *2009 24th Annual IEEE Conference on Computational Complexity*, pages 229–242, 2009.

[Aar16] Scott Aaronson. The complexity of quantum states and transformations: From quantum money to black holes, 2016.

[Aar18] Scott Aaronson. Shadow tomography of quantum states. *SIAM Journal on Computing*, 49(5):STOC18–368–STOC18–394, 2018.

[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, page 41–60, New York, NY, USA, 2012. Association for Computing Machinery.

[AK07] Scott Aaronson and Greg Kuperberg. Quantum versus classical proofs and advice. In *Twenty-Second Annual IEEE Conference on Computational Complexity (CCC'07)*, pages 115–128. IEEE, 2007.

[AKL$^+$22] Prabhanjan Ananth, Fatih Kaleoglu, Xingjian Li, Qipeng Liu, and Mark Zhandry. On the feasibility of unclonable encryption, and more. Cryptology ePrint Archive, Paper 2022/884, 2022. https://eprint.iacr.org/2022/884.

[ALL$^+$20] Scott Aaronson, Jiahui Liu, Qipeng Liu, Mark Zhandry, and Ruizhe Zhang. New approaches for quantum copy-protection, 2020.

[ALL$^+$21] Scott Aaronson, Jiahui Liu, Qipeng Liu, Mark Zhandry, and Ruizhe Zhang. New approaches for quantum copy-protection. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021*, pages 526–555, Cham, 2021. Springer International Publishing.

[ALP21] Prabhanjan Ananth and Rolando L. La Placa. Secure software leasing. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021*, pages 501–530, Cham, 2021. Springer International Publishing.

[BBBV97] Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM journal on Computing*, 26(5):1510–1523, 1997.

[BGG$^+$23] James Bartusek, Sanjam Garg, Vipul Goyal, Dakshita Khurana, Giulio Malavolta, Justin Raizes, and Bhaskar Roberts. Obfuscation and outsourced computation with certified deletion. Cryptology ePrint Archive, Paper 2023/265, 2023. https://eprint.iacr.org/2023/265.

[BGHD$^+$23] Khashayar Barooti, Alex B. Grilo, Loïs Huguenin-Dumittan, Giulio Malavolta, Or Sattath, Quoc-Huy Vu, and Michael Walter. Public-key encryption with quantum keys. In Guy Rothblum and Hoeteck Wee, editors, *Theory of Cryptography*, pages 198–227, Cham, 2023. Springer Nature Switzerland.

[BV18] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *J. ACM*, 65(6), nov 2018.

[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*, pages 280–300. Springer, 2013.

[CGJS15] Nishanth Chandran, Vipul Goyal, Aayush Jain, and Amit Sahai. Functional encryption: Decentralised and delegatable. Cryptology ePrint Archive, Paper 2015/1017, 2015. https://eprint.iacr.org/2015/1017.

[ÇGLZR24] Alper Çakan, Vipul Goyal, Chen-Da Liu-Zhang, and João Ribeiro. Unbounded leakage-resilience and leakage-detection in a quantum world. In *Theory of Cryptography*. Springer International Publishing, 2024. https://eprint.iacr.org/2023/410.

[CLLZ21] Andrea Coladangelo, Jiahui Liu, Qipeng Liu, and Mark Zhandry. Hidden cosets and applications to unclonable cryptography. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021*, pages 556–584, Cham, 2021. Springer International Publishing.

[CMP20] Andrea Coladangelo, Christian Majenz, and Alexander Poremba. Quantum copy-protection of compute-and-compare programs in the quantum random oracle model. Cryptology ePrint Archive, Paper 2020/1194, 2020. https://eprint.iacr.org/2020/1194.

[CV22] Eric Culf and Thomas Vidick. A monogamy-of-entanglement game for subspace coset states. *Quantum*, 6:791, 2022.

[CZDC19] Yu Chen, Jiang Zhang, Yi Deng, and Jinyong Chang. Kdm security for identity-based encryption: Constructions and separations. *Information Sciences*, 486:450–473, 2019.

[FGH$^+$12] Edward Farhi, David Gosset, Avinatan Hassidim, Andrew Lutomirski, and Peter Shor. Quantum money from knots. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, page 276–289, New York, NY, USA, 2012. Association for Computing Machinery.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, aug 1986.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98, 2006.

[GZ20] Marios Georgiou and Mark Zhandry. Unclonable decryption keys. Cryptology ePrint Archive, Paper 2020/877, 2020. https://eprint.iacr.org/2020/877.

[KN22] Fuyuki Kitagawa and Ryo Nishimaki. Functional encryption with secure key leasing. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022*, pages 569–598, Cham, 2022. Springer Nature Switzerland.

[KNT22] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Obfustopia built on secret-key functional encryption. volume 35, page 19, Jun 2022.

[KNY21] Fuyuki Kitagawa, Ryo Nishimaki, and Takashi Yamakawa. Secure software leasing from standard assumptions. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography*, pages 31–61, Cham, 2021. Springer International Publishing.

[Kre21] William Kretschmer. Quantum pseudorandomness and classical complexity. In *16th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[KT24] Dakshita Khurana and Kabir Tomer. Commitments from quantum one-wayness, 2024.

[LAF$^+$09] Andrew Lutomirski, Scott Aaronson, Edward Farhi, David Gosset, Avinatan Hassidim, Jonathan Kelner, and Peter Shor. Breaking and making quantum money: toward a new quantum cryptographic protocol, 2009.

[LLQZ22] Jiahui Liu, Qipeng Liu, Luowen Qian, and Mark Zhandry. Collusion resistant copy-protection for watermarkable functionalities. Cryptology ePrint Archive, Paper 2022/1429, 2022. https://eprint.iacr.org/2022/1429.

[MW04] C. Marriott and J. Watrous. Quantum arthur-merlin games. In *Proceedings. 19th IEEE Annual Conference on Computational Complexity, 2004.*, pages 275–285, 2004.

[NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[OW16] Ryan O'Donnell and John Wright. Efficient quantum tomography. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 899–912, 2016.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In George Robert Blakley and David Chaum, editors, *Advances in Cryptology*, pages 47–53, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, pages 457–473, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[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*, STOC '14, page 475–484, New York, NY, USA, 2014. Association for Computing Machinery.

[SW22] Or Sattath and Shai Wyborski. Uncloneable decryptors from quantum copy-protection, 2022.

[VZ21] Thomas Vidick and Tina Zhang. Classical proofs of quantum knowledge. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021*, pages 630–660, Cham, 2021. Springer International Publishing.

[Wat18] John Watrous. *The Theory of Quantum Information.* Cambridge University Press, 2018.

[Wie83] Stephen Wiesner. Conjugate coding. *SIGACT News*, 15(1):78–88, jan 1983.

[Wil15] Mark Wilde. Quantum information theory lecture notes, lecture 16, 2015.

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under lwe. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 600–611, 2017.

[YAL+19] Rupeng Yang, Man Ho Au, Junzuo Lai, Qiuliang Xu, and Zuoxia Yu. Collusion resistant watermarking schemes for cryptographic functionalities. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019*, pages 371–398, Cham, 2019. Springer International Publishing.

[Zha12a] Mark Zhandry. How to construct quantum random functions. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 679–687, 2012.

[Zha12b] Mark Zhandry. Secure identity-based encryption in the quantum random oracle model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pages 758–775, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[Zha20] Mark Zhandry. Schrödinger's pirate: How to trace a quantum decoder. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography*, pages 61–91, Cham, 2020. Springer International Publishing.

A Proofs from Section 3

A.1 Proof of Theorem 8

We can write ρ as $\rho = \sum_{j,k} \alpha_{j,k} |j\rangle\langle j| \otimes |k\rangle\langle k|$. Then, $(M_i \otimes I)\rho(M_i^\dagger \otimes I) = \sum_{j,k} \alpha_{j,k}(M_i|j\rangle\langle j|M_i^\dagger) \otimes |k\rangle\langle k|$ and therefore $(\mathrm{Tr} \otimes I)(M_i \otimes I)\rho(M_i^\dagger \otimes I) = \sum_{j,k} \alpha_{j,k}\langle j|M_i^\dagger M_i|j\rangle \otimes |k\rangle\langle k|$. Note that this summation only depends on the POVM element $M_i^\dagger M_i$. The same argument applies to Λ'. Hence, the result follows by POVM equivalence of Λ, Λ'.

A.2 Proof of Lemma 3

Let $\|\cdot\|_1$ be the trace norm, and we have $\|\rho - \sigma\|_{Tr} = \frac{1}{2}\|\rho - \sigma\|_1$. Define $q_i = \operatorname{Tr}\left\{M_i \sigma M_i^\dagger\right\}$ and we also have $p_i = \operatorname{Tr}\left\{M_i \sigma M_i^\dagger\right\}$. Then,

$$\begin{aligned}\left\|p_i\rho' - q_i\sigma'\right\|_1 &= \left\|p_i(\rho' - \sigma') - (q_i - p_i)\sigma'\right\|_1 \\ &\geq \left|p_i\left\|(\rho' - \sigma')\right\|_1 - |q_i - p_i|\left\|\sigma'\right\|_1\right| \\ &\geq p_i\left\|(\rho' - \sigma')\right\|_1 - \varepsilon\end{aligned}$$

Last part follows from $|q_i - p_i| \leq \|\rho - \sigma\|_{Tr} \leq \varepsilon$ ad $\|\sigma'\|_1 = 1$. We also have by Schatten norm duality

$$\begin{aligned}\left\|p_i\rho' - q_i\sigma'\right\|_1 &= \left\|M_i(\rho - \sigma)M_i^\dagger\right\|_1 \\ &= \sup_{-\mathbb{I} \leq E \leq \mathbb{I}} \operatorname{Tr}\left\{EM_i(\rho - \sigma)M_i^\dagger\right\} \\ &= \sup_{-\mathbb{I} \leq E \leq \mathbb{I}} \operatorname{Tr}\left\{M_i^\dagger EM_i(\rho - \sigma)\right\} \\ &\leq \sup_{-\mathbb{I} \leq E \leq \mathbb{I}} \operatorname{Tr}\{E(\rho - \sigma)\} \\ &= \|\rho - \sigma\|_1 \leq 2\varepsilon.\end{aligned}$$

Above we also used the fact that when $-\mathbb{I} \leq E \leq \mathbb{I}$, we also have $-\mathbb{I} \leq M_i^\dagger EM_i \leq \mathbb{I}$. This is because $M_i^\dagger EM_i$ is positive semidefinite and $\langle v|(I - M_i^\dagger EM_i)|v\rangle = \langle v|v\rangle - \langle v|M_i^\dagger EM_i|v\rangle \geq \langle v|v\rangle - \langle v|M_i^\dagger M_i|v\rangle \geq 0$ since $\langle v|M_i^\dagger EM_i|v\rangle \leq \langle v|M_i^\dagger M_i|v\rangle$ and $\sum_i M_i^\dagger M_i = \mathbb{I}$.

Combining the above yields the result.

A.3 Proof of Theorem 9

First, we will prove the case where $\varepsilon = 0$. We will prove it only for pure states, but the general case follows from purification. Let $|\psi\rangle$ be any state of appropriate dimension. We can write $|\psi\rangle = \sum_{j \in \mathcal{J}, k \in \mathcal{K}} \alpha_{j,k}|v_j\rangle \otimes |w_k\rangle$ where $\{|v_j\rangle\}_{j \in \mathcal{J}}, \{|w_k\rangle\}_{k \in \mathcal{K}}$ are orthonormal eigenbases of Π_1 and Π_1' respectively. We have $\Pi_1 = \sum_{j \in \mathcal{J}'}|v_j\rangle\langle v_j|$ and $\Pi_1' = \sum_{k \in \mathcal{K}'}|w_k\rangle\langle w_k|$ for some subsets $\mathcal{J}' \subseteq \mathcal{J}, \mathcal{K}' \subseteq \mathcal{K}$. Since $\operatorname{Tr}\{\Pi_1 \otimes \Pi_1'\rho\} = 1$, we get $\alpha_{j,k} = 0$ if $(j,k) \notin \mathcal{J}' \times \mathcal{K}'$.

We can write the post-measurement state conditioned on outcome i as $|\phi\rangle/\||\phi\rangle\|$ where we define the subnormalized state $|\phi\rangle = \sum_{j \in \mathcal{J}', k \in \mathcal{K}'} \alpha_{j,k}(M_i|v_j\rangle) \otimes |w_k\rangle$. When we apply $I \otimes \Pi_1'$ to $|\phi\rangle$, we get $|\phi\rangle$ again. Hence, $\operatorname{Tr}\left\{\Pi_1' \frac{|\phi\rangle\langle\phi|}{\||\phi\rangle\|^2}\right\} = 1$, completing the first part of the proof.

Now, we move onto any $\varepsilon \in (0,1]$. Let ρ' denote the post-measurement state obtained after applying $\Lambda \otimes \Lambda'$ to ρ and obtaining the outcome $(1,1)$. Note that ρ' satisfies the claim with $\varepsilon = 0$ since the measurement $\Lambda \otimes \Lambda'$ is projective. Then, by Lemma 2, $\|\rho - \rho'\|_{Tr} \leq \sqrt{\varepsilon}$ since canonical square root implementation of a projective measurement is the original measurement itself. Hence, applying the measurement M on the first register and conditioning the outcome i, the post-measurement states of the second registers will have trace distance at most $3\sqrt{\varepsilon}/2p_i$ by Lemma 3. Hence, invoking the sub-claim for ρ' with $\varepsilon = 0$ and using the trace distance bound, we get the result.

A.4 Proof of Theorem 10

We will prove the result only for pure states $\rho = |\phi\rangle\langle\phi|$ and the general case follows from convexity.

We will first prove the case $n = 2$. Since Π_1, Π_2 are commuting projectors, there exists an orthonormal basis $\{|v_i\rangle\}_{i\in\mathcal{I}}$ and $\mathcal{I}_1, \mathcal{I}_2 \subseteq \mathcal{I}$ such that $\Pi_1 = \sum_{i\in\mathcal{I}_1}|v_i\rangle\langle v_i|$ and $\Pi_2 = \sum_{i\in\mathcal{I}_2}|v_i\rangle\langle v_i|$. We also have $|\phi\rangle = \sum_{i\in\mathcal{I}} c_i|v_i\rangle$ for some $\{c_i\}_{i\in\mathcal{I}}$ with $\sum_{i\in\mathcal{I}}|c_i|^2 = 1$. Then,

$$\begin{aligned}
\mathrm{Tr}[\Pi_1\rho] + \mathrm{Tr}[\Pi_2\rho] - \mathrm{Tr}[(I - \Pi_1\Pi_2)\rho] &= \sum_{i\in\mathcal{I}_1}|c_i|^2 + \sum_{i\in\mathcal{I}_2}|c_i|^2 - \sum_{i\in\mathcal{I}_1\cap\mathcal{I}_2}|c_i|^2 \\
&= \sum_{i\in\mathcal{I}_1\cup\mathcal{I}_2}|c_i|^2 \\
&\leq \sum_{i\in\mathcal{I}}|c_i|^2 = 1.
\end{aligned}$$

Hence, $\mathrm{Tr}[(I - \Pi_1\Pi_2)\rho] \leq \mathrm{Tr}[(I - \Pi_1)\rho] + \mathrm{Tr}[(I - \Pi_2)\rho]$. The general case follows by repeatedly applying this case and observing that Π_i commutes with $\Pi_{i+1}\cdots\Pi_n$.

A.5 Proof of Theorem 16

First, we note that the result does not directly follow from the efficiency of $\mathsf{API}^{\varepsilon,\delta}$. The reason is that while it is indeed efficient, it obtains superpositions of (exponentially many) outputs from the underlying distributions.

The proof will closely follow the proof of [Zha20, Theorem 6.5]. We first state some of technical results that will be needed in the proof.

A.5.1 Technical Lemmata

Lemma 13. *Let $\mathcal{D}_0, \mathcal{D}_1$ be two efficient distributions with the same support and $\mathcal{P}$ be a collection of projective measurements indexed by this support. Suppose $\mathcal{D}_0 \equiv \mathcal{D}_1$. Then, $\mathsf{API}^{\varepsilon,\delta}_{\mathcal{P},\mathcal{D}_0}\rho \equiv \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P},\mathcal{D}_1}\rho$ for any state ρ of appropriate dimension.*

While the claim might seem obvious, it needs to be proven formally since $\mathsf{API}^{\varepsilon,\delta}_{\mathcal{P},\mathcal{D}}$ does not work by obtaining random samples from $\mathcal{D}$ but instead runs the algorithm $\mathcal{D}$ on various choices on random coins.

Proof. When we inspect the actual implementation of $\mathsf{API}^{\varepsilon,\delta}$ and proof of [Zha20, Theorem 6.2], we see that the output distribution of $\mathsf{API}^{\varepsilon,\delta}_{\mathcal{P},\mathcal{D}}\rho$ is equivalent to the following:

1. Sample $p \leftarrow \mathsf{PI}_{\mathcal{P}_\mathcal{D}}\rho$.
2. Flip $2T$ independent biased coins, where each coin has expected value p.
3. Output some deterministic function of all coin flips.

Since $\mathcal{D}_0 \equiv \mathcal{D}_1$ implies $\mathsf{PI}(\mathcal{P}_{\mathcal{D}_0}) = \mathsf{PI}(\mathcal{P}_{\mathcal{D}_1})$, the result follows. □

Lemma 14. *Let $\mathcal{D}_0, \mathcal{D}_1$ be two distributions with sampling time $p(\lambda)$ such that $\mathcal{D}_0 \approx^c_{\nu(\lambda)} \mathcal{D}_1$ for all adversaries that run in time $t(\lambda)$. Define $\mathcal{D}_b^{s(\lambda)}$ to be the distribution where we sample s independent samples from $\mathcal{D}_b$. Then, $\mathcal{D}_0^s \approx^c_{s(\lambda)\cdot\nu(\lambda)} \mathcal{D}_1^s$ for all adversaries that run in time $t(\lambda)/(p(\lambda)\cdot s(\lambda))$.*

Proof. The result follows from a standard hybrid argument. We give in full detail for completeness.

For all $i \in \{0, 1, \dots, s(\lambda)\}$, we define the hybrid distribution Hyb_i as the distribution where the first i components are sampled from $\mathcal{D}_1$ and the rest are sampled from $\mathcal{D}_0$. Observe that Hyb_0 is $\mathcal{D}_0^s$ and Hyb_s is $\mathcal{D}_1^s$.

Now, we claim $\mathsf{Hyb}_{i-1} \approx^c_{\nu(\lambda)} \mathsf{Hyb}_i$ for adversaries that run in $t(\lambda)/s(\lambda)$, for all $i \in [s]$. Suppose otherwise, for a contradiction. Let $\mathcal{A}$ be the that distinguishes them. Then, we can create an adversary $\mathcal{A}'$ for distinguishing $\mathcal{D}_0$ versus $\mathcal{D}_1$ as follows.

$\underline{\mathcal{A}'(a)}$

1. For $j \in [s]$, sample $a_j \leftarrow \mathcal{D}_1$ if $j \leq i - 1$ and $a_j \leftarrow \mathcal{D}_0$ if $j > i$.
2. Set $a_i = a$.
3. Output $\mathcal{A}((a_j)_{j \in [s]})$.

It is easy to see that $\mathcal{A}'$ runs in time $O(t(\lambda)/(p(\lambda) \cdot s(\lambda)) \cdot p(\lambda) \cdot s(\lambda))$ and has advantage $\geq \nu(\lambda)$, which is a contradiction.

Finally, triangle inequality yields the claim. □

Lemma 15 ([Zha20])**.** *Let A be a set. Sample Π to be a random permutation on A. Sample random functions $G : [s] \to A$ and $F : A \to [s]$. Then, for any quantum algorithm B making Q quantum queries to its oracle, we have*

$$\left|\Pr\left[B^{\Pi}() = 1\right] - \Pr\left[B^{G \circ F}() = 1\right]\right| \leq O(Q^3/s + Q^3/|A|).$$

Lemma 16 ([Zha12b])**.** *Sample a random function $F : \mathcal{A} \to \mathcal{B}$ and a $2Q$-wise independent function $E : \mathcal{A} \to \mathcal{B}$. Then, for any quantum algorithm B making Q quantum queries to its oracle, we have*

$$\Pr\left[B^F() = 1\right] = \Pr\left[B^E() = 1\right].$$

A.5.2 Proof of the Theorem

Now we move onto the proof of the theorem. We will construct a sequence of hybrid distributions, starting with $\vec{p_0}$ and ending with $\vec{p_1}$, that are obtained by modifying the previous one. Without loss of generality, assume that all $\mathcal{B}_\ell^b$ have the same random coin set $\mathcal{R}$. Let s be a parameter that we will set later. We assume that $|\mathcal{R}|$ is at least s, which is without loss of generality since we can pad the random coins (and later ignore the padding when using the coins).

Hyb_0: Same as $\vec{p_0}$.

Hyb_1: We now sample ρ, pp as $\mathcal{S}^1(1^\lambda)$.

Hyb_2: For all $\ell \in [k]$, sample a random permutation $\Pi_\ell : \mathcal{R} \to \mathcal{R}$. Then, instead of applying $\bigotimes_{\ell \in [k]} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell, \mathcal{B}_\ell^0(pp)}$ to ρ, now apply $\bigotimes_{\ell \in [k]} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell, \mathcal{B}_\ell'^0(pp))}$ where we define $\mathcal{B}_\ell'^0(pp; r) = \mathcal{B}_\ell^0(pp; \Pi_\ell(r))$.

$\mathsf{Hyb}_{3,i}$ for $i \in [k]$: Sample random functions $G_\ell : [s] \to \mathcal{R}$ and $F_\ell : \mathcal{R} \to [s]$ for all $\ell \in [i]$. We now apply $\left(\bigotimes_{\ell \in [i]} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell, \mathcal{B}_\ell''^0(pp))}\right) \otimes \left(\bigotimes_{\ell \in \{i+1,\dots,k\}} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell, \mathcal{B}_\ell'^0(pp))}\right)$ where we define $\mathcal{B}_\ell''^0(pp; r) = \mathcal{B}_\ell^0(pp; G(F(r)))$.

$\mathsf{Hyb}_{4,i}$ **for** $i \in [k]$**:** Sample $2Q$-wise independent function $E_\ell : \mathcal{R} \to [s]$ for all $\ell \in [i]$. We now apply $\left(\bigotimes_{\ell\in[i]} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{B}_\ell'''^0(pp))}\right) \otimes \left(\bigotimes_{\ell\in\{i+1,\dots,k\}} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{B}_\ell''^0(pp))}\right)$ where we define $\mathcal{B}_\ell'''^0(pp;r) = \mathcal{B}_\ell^0(pp;G(E(r)))$.

$\mathsf{Hyb}_{5,i}$ **for** $i \in \{0,1,\dots,k-1\}$**:** We now apply $\left(\bigotimes_{\ell\in[i]} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{B}_\ell'''^1(pp))}\right) \otimes \left(\bigotimes_{\ell\in\{i+1,\dots,k\}} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{B}_\ell'''^0(pp))}\right)$.

$\mathsf{Hyb}_{6,k-i+1}$ **for** $i \in [k]$**:** We now apply $\left(\bigotimes_{\ell\in[i]} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{B}_\ell'''^1(pp))}\right) \otimes \left(\bigotimes_{\ell\in\{i+1,\dots,k\}} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{B}_\ell''^1(pp))}\right)$ where we define $\mathcal{B}_\ell'''^1(pp;r) = \mathcal{B}_\ell^1(pp;G(E(r)))$ and $\mathcal{B}_\ell''^1(pp;r) = \mathcal{B}_\ell^1(pp;G(F(r)))$.

$\mathsf{Hyb}_{7,k-i+1}$ **for** $i \in [k]$**:** We now apply $\left(\bigotimes_{\ell\in[i]} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{B}_\ell''^1(pp))}\right) \otimes \left(\bigotimes_{\ell\in\{i+1,\dots,k\}} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{B}_\ell'^1(pp))}\right)$ where we define $\mathcal{B}_\ell'^1(pp;r) = \mathcal{B}_\ell^1(pp;\Pi_\ell(r))$.

Hyb_8**:** We now apply $\bigotimes_{\ell\in[k]} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{B}_\ell'^1(pp))}$ to ρ.

Hyb_9**:** Same as $\vec{p_1}$.

Lemma 17. $|\mathsf{Hyb}_0 - \mathsf{Hyb}_1| \leq \nu(\lambda)$

Lemma 18. $\mathsf{Hyb}_1 \equiv \mathsf{Hyb}_2$ *and* $\mathsf{Hyb}_8 \equiv \mathsf{Hyb}_9$.

Proof. Observe that $\mathcal{B}_\ell'^b(pp)$ and $\mathcal{B}_\ell^b(pp)$ are exactly the same distribution. The result follows from Lemma 13. □

Lemma 19.

- $|\mathsf{Hyb}_2 - \mathsf{Hyb}_{3,1}| \leq O(Q^3/s)$.
- $|\mathsf{Hyb}_{3,i} - \mathsf{Hyb}_{3,i+1}| \leq O(Q^3/s)$ *for all* $i \in [k-1]$.
- $|\mathsf{Hyb}_{7,i} - \mathsf{Hyb}_{7,i+1}| \leq O(Q^3/s)$ *for all* $i \in [k-1]$.
- $|\mathsf{Hyb}_{7,k} - \mathsf{Hyb}_8| \leq O(Q^3/s)$.

Proof. Follows from Lemma 15. □

Lemma 20.

- $\mathsf{Hyb}_{3,k} \equiv \mathsf{Hyb}_{4,1}$.
- $\mathsf{Hyb}_{4,i} \equiv \mathsf{Hyb}_{4,i+1}$ *for all* $i \in [k-1]$.
- $\mathsf{Hyb}_{6,k} \equiv \mathsf{Hyb}_{7,1}$.
- $\mathsf{Hyb}_{6,i} \equiv \mathsf{Hyb}_{6,i+1}$ *for all* $i \in [k-1]$.

Proof. Follows from Lemma 16. □

Lemma 21.

- $|\mathsf{Hyb}_{5,k-1} - \mathsf{Hyb}_{6,0}| \leq \nu(\lambda) \cdot s(\lambda)$.
- $|\mathsf{Hyb}_{5,i-1} - \mathsf{Hyb}_{5,i}| \leq \nu(\lambda) \cdot s(\lambda)$ *for* $i \in [k]$

Proof. Observe that $\mathcal{B}^b_\ell(pp; G(E(\cdot)))$ can be interpreted as s samples from $\mathcal{B}^b$ with the input selecting which sample to use. Also, both experiments can be computed in time $\mathsf{poly}(\lambda) \cdot k \cdot s$. The result then follows from Lemma 14. □

Combining the above, we get $|\vec{p_0} - \vec{p_1}| < O(k \cdot (Q^3/s + \nu(\lambda) \cdot s(\lambda)))$. We set $s = 1/\mu(\lambda)$, which yields the result since $Q = \mathsf{poly}(\lambda)$.

A.6 Proofs of Theorem 18, Theorem 19,Theorem 20

Proof of Theorem 18 See [ALL⁺20, Corollary 3] for the proofs of the first two points. Note that while they consider the same threshold value η_ℓ for all indices ℓ, an inspection of their proof easily shows that the results still hold for any η_ℓ.

Combining the first two bullet points yields the third bullet point in a straightforward manner. Fourth point follows similarly to arguments below for Theorem 19.

Proof of Theorem 19 We will only prove the first claim, and the second prove follows by the same argument.

Fix any $\ell \in [k]$. Consider the projective measurement $\mathcal{M}_\ell$ where we apply $\mathsf{PI}(\mathcal{P}_{\ell\mathcal{D}_\ell})$ to the ℓ-th register and apply I to the other registers. Then, we have $\Pr[(\mathcal{M}_\ell \rho) \leq (\vec{p})_\ell + 2\varepsilon] \geq 1 - 2\delta$ since $\mathsf{API}^{\varepsilon,\delta}$ is (ε, δ)-almost projective and since it δ-approximates PI in ε-shift distance. Note while ρ' is obtained after a measurement on all registers, we can assume that $\mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{D}_\ell}$ was applied last since measurements on disjoint registers commute.

Now, observe that $\mathcal{M}_1\mathcal{M}_2 \cdots \mathcal{M}_k = \left(\bigotimes_{\ell \in [k]} \mathsf{PI}(\mathcal{P}_{\ell\mathcal{D}_\ell})\right)$ and that $\mathcal{M}_\ell$ commute. We define $\mathcal{M}'_\ell$ to be the binary projective measurement where we apply $\mathcal{M}_\ell$ and output 1 if the outcome is $\leq \vec{p}_\ell + 2\varepsilon$. Then, by above we have $\Pr[\mathcal{M}'_\ell \rho' = 1] \geq 1 - 2\delta$. We get $\Pr\left[\forall \ell \in [k] \quad (\vec{p'})_\ell \leq (\vec{p})_\ell + 2\varepsilon\right] \geq 1 - 2 \cdot k \cdot \delta$ by Theorem 10.

A.7 Proof of Theorem 20

Follows similarly to arguments above for Theorem 19.

B Connections of the Signature Scheme to the Public-key Encryption Scheme

The only difference between the scheme in Hyb_5 in the proof of Section 9.5 and our PKE scheme is the hidden trigger mechanism and the associated extra programs and keys, $K, K_3, K_4, \mathsf{OPEval}, \mathsf{OPVer}$, which is independent of the actual encryption mechanism and can be generated by the adversary itself during the reduction to the security of PKE scheme; and the fact that the ciphertext programs OQ now include two branches along with a mode parameter. The mode $\mathsf{mode} = \mathsf{eval}$ is the same as the original ciphertext program of our PKE scheme, and $\mathsf{mode} = \mathsf{verify}$ is a point function that checks if the input has the same image as the encrypted message under a one-way function f. However, we can see that the scheme is still secure with this addition.

First argument is as follows. We can first invoke the security argument (Section 7.3) without $\mathsf{mode} = \mathsf{verify}$, and show that any adversary can win the game (i.e. correctly predict the encrypted messages) with subexponential probability. Then, adding back the verify mode, we can consider the copy-protected keys obtained by the adversary as quantum auxiliary information, and we can replace the verify mode with a compute-and-compare obfuscation. Since by the security of the scheme without verify mode we have that the adversary cannot predict the messages correctly given its auxiliary information, we can finally invoke the security of the compute-and-compare obfuscation to conclude that the adversary still cannot predict the messages correctly given verify mode.

An alternative way of concluding the security of the scheme is by repeating the security proof in Section 7.3. It is easy to see that the only part that will be affected by verify mode is the final extraction argument using compute-and-compare obfuscation. However, it is easy to see that by the security of the injective one-way function f that the compute-and-compare obfuscation arguments still works. In this case, we will modify our compute-and-compare programs so that the compute program also accepts a mode parameter, and either performs coset vector verification or compares the image of the input to $f(m)$. However, any adversary that can succesfully find an input that passes the compare condition must have correct coset vectors, since it cannot have a correct preimage for $f(m)$ by the security of the one-way function f. Thus, the security proof in Section 7.3 still works.

C Proof of Anti-Piracy Security of the General Copy-Protection Scheme

In this section, we prove Theorem 42.

Define Hyb_0 to the original anti-piracy game. Define Hyb_1 by modifying Hyb_0 by changing the way we sample the identity strings during each quantum copy-protected function generation as follows. Let the challenger record each sampled identity when answering each query, and when answering a new query, it samples uniformly at random an identity value from the set $\{1, \ldots, 2^\lambda - 1\}$ *that has not appeared before*. That is, we sample unique identity strings for each query to QGen. Also, we define the following notation. Let $id_{\alpha(i)}$ be the i^{th} value sampled where $\alpha(\cdot)$ is the permutation $[k] \to [k]$ such that $0 < id_1 < \cdots < id_k < 2^\lambda$. That is, $id_{\alpha(i)}$ is the identity string that is sampling during the i^{th} query of the adversary. For simplicity of notation, we also set $id_0 = 0$ and $id_{k+1} = 2^\lambda$.

Define Hyb_2 by modifying Hyb_1 as follows. At the end of the game, instead of using threshold implementations $\mathsf{TI}_{\mathcal{D},\gamma}^{\mathcal{O}}$, the challenger uses approximate threshold implementations $\mathsf{ATI}_{\mathcal{D},\frac{31\gamma}{32}}^{\varepsilon,\delta,\mathcal{O}}$ with $\varepsilon = \frac{\gamma}{32k}$ and $\delta = 2^{-4\lambda}$. It outputs 1 if and only if all ATI output 1.

Claim 30. $\Pr[\mathsf{Hyb}_2 = 1] > \Pr[\mathsf{Hyb}_0 = 1] - \exp(-\lambda)$.

Proof. Follows from the same argument as in Section 7.3. □

Therefore, $\mathcal{A}$ wins in Hyb_2 with probability $\frac{1}{p(\cdot)}$ for some polynomial $p(\cdot)$ and infinitely many values of $\lambda > 0$. Note that in Hyb_2, now the challenger is also efficient by Theorem 17 and our choice of ε, δ.

We define the following notation and the monogamy-of-entanglement type game.

Definition 50. *For all $j \in [k]$, let $(A_i^j, s_i^j, s_i'^j)_{i \in [c_L(\lambda)]}$ denote the tuple of subspaces and vectors sampled during the sampling of the $(\alpha^{-1}(j))$-th key. That is, it is the coset tuple associated with id_j.*

$\underline{\mathcal{G}(\lambda, (\mathcal{A}_0', \mathcal{A}_1', \mathcal{A}_2'))}$

1. The challenger instantiates the copy-protection scheme as $sk \leftarrow \mathsf{Setup}(1^\lambda)$.

2. The challenger samples a function f from $\mathcal{F}$.
3. For multiple rounds, $\mathcal{A}$ makes quantum key queries. For each query, the challenger generates a key as $\mathsf{R} \leftarrow \mathsf{QGen}(sk, f)$ and submits R to the adversary.
4. The adversary outputs a *bipartite* register $\mathsf{R}_{\mathsf{bip}}$ and an index $j^* \in [k]$, where k is the number of queries it made.
5. For $\ell \in \{1, 2\}$, the challenger does the following.
   5.1. Sample $r_\ell \leftarrow \mathcal{D}$.
   5.2. Run $\mathcal{A}'_\ell$ on $\mathsf{R}_{\mathsf{bip}}[\ell]$, $(A_i^{j^*})_{i\in[m(\lambda)]}$ and r_ℓ to obtain a tuple of vectors $(v_{\ell,i})_{i\in[m(\lambda)]}$.
   5.3. For all $i \in [m(\lambda)]$, check if $v_{\ell,i} \in A_i^{j^*} + s_i^{j^*}$ if $(r_\ell)_i = 0$ and check if $v_{\ell,i} \in (A^{j^*})_i^{\perp} + s_i'^{j^*}$ if $(r_\ell)_i = 1$.

   If all the checks pass, the challenger outputs 1. Otherwise, it outputs 0.

It is easy to see that an adversary $\mathcal{A}'$ that wins $\mathcal{G}$ gives us a contradiction for Theorem 24, since using $\mathcal{A}'$ we can give an adversary for the monogamy-of-entanglement game, which simply simulates $\mathcal{A}'$ playing $\mathcal{G}$ by sampling the PRF key K_1, the extra oracles and so on efficiently itself. In particular note that the oracle $\mathcal{O}_f$ in $\mathcal{G}$ can be implemented using PMem from Theorem 24[48].

Now, we construct a tuple of adversaries $(\mathcal{A}'_0, \mathcal{A}'_1, \mathcal{A}'_2)$ for $\mathcal{G}$, starting with $\mathcal{A}'_0$. We note that from now on, whenever we are considering the freeloaders, we assume that they no longer have access to the oracle $\mathcal{O}_f$ that $\mathcal{A}$ had during the query phase, and each will have access to some modified oracle that will be clear from context.

Let $\mathcal{O}_j$ for $j \in \{0, \ldots, k+1\}$ be efficient oracles, which we will define later. Define $\mathcal{A}'_0$ as follows.

$\underline{\mathcal{A}'_0(1^\lambda)}$

1. Simulate $\mathcal{A}$ by interacting with Samp and the challenger, making a query to QGen whenever $\mathcal{A}$ makes a query, and forwarding the obtained program to it. Let f be the function determined at the end of the setup phase and let $\mathsf{R}_{\mathsf{adv}}$ be the $(k+1)$-partite register (with state σ) output by $\mathcal{A}$ at the output phase.
2. Uniformly at random sample x, y, j^* such that $1 \leq x < y \leq k+1$ and $j^* \in \{1, \ldots, k\}$.
3. Apply $\mathsf{API}_{\mathcal{D}}^{\varepsilon,\delta,\mathcal{O}_0}$ to all registers $\mathsf{R}_{\mathsf{adv}}[\ell]$ for $\ell \in [k+1]$, let $b_{\ell,0}$ be the measurement outcomes.
4. Apply $\mathsf{API}_{\mathcal{D}}^{\varepsilon,\delta,\mathcal{O}_i}$ in succession for $i = 1$ to j^* to $\mathsf{R}_{\mathsf{adv}}[x]$, let $b_{x,i}$ be the measurement outcomes.
5. Apply $\mathsf{API}_{\mathcal{D}}^{\varepsilon,\delta,\mathcal{O}_i}$ in succession for $i = 1$ to j^* to $\mathsf{R}_{\mathsf{adv}}[y]$, let $b_{y,i}$ be the measurement outcomes.

[48] While the original theorem statement therein uses indistinguishability obfuscation, it is trivial to see that the result still holds when we instead use ideal oracles.

```
6. Output
   $$((\mathsf{R}_{\mathsf{adv}}[x], j^*, x, y, (b_{\ell,0})_{\ell\in[k+1]}, (b_{x,i})_{i\in[j^*]}, (b_{y,i})_{i\in[j^*]}),$$
   $$(\mathsf{R}_{\mathsf{adv}}[y], j^*, x, y, (b_{\ell,0})_{\ell\in[k+1]}, (b_{x,i})_{i\in[j^*]}, (b_{y,i})_{i\in[j^*]},),$$
   $$j^*).$$
```

For $j \in \{1, \ldots, k\}$, define $\mathcal{O}_j$ to be the following oracles.

```
$\mathcal{O}_j(id, x, (v_i)_{i\in[m(\lambda)]})$
  Hardcoded: $K_1, f, id_j$
  1. $(A_i, s_i, s'_i)_{i\in[m(\lambda)]} \leftarrow \mathsf{CosetGen}(1^\lambda, m(\lambda), \lambda; F_1(K_1, id))$.
  2. If $id < id_j$, output $\perp$ and terminate.
  3. For each $i \in [m(\lambda)]$, check if $v_i \in A_i + s_i$ if $(x)_i = 0$ and check if $v_i \in A_i^\perp + s'_i$ if $(x)_i = 1$.
     If any of the checks fail, output $\perp$ and terminate.
  4. Output $f(x)$.
```

We define $\mathcal{O}_0$ to be the original oracle $\mathcal{O}_f$ and we define $\mathcal{O}_{k+1}$ to be the empty oracle that always outputs $\perp$. We also define some intermediary oracles. Define the following for all $j \in \{0, 1, \ldots, k\}$ and $\Delta \in \{0, 1, \ldots, id_{j+1} - id_j - 1\}$. For notational convenience, also define $\mathcal{O}_{j, id_{j+1} - id_j}$ to be $\mathcal{O}_{j+1,0}$ for all $j \in \{0, 1, \ldots, k\}$. Also note that $\mathcal{O}_{0,0}$ is exactly the same as $\mathcal{O}_0$ for $j \in [k]$.

```
$\mathcal{O}_{j,\Delta}(id, x, (v_i)_{i\in[m(\lambda)]})$
  Hardcoded: $K_1, f, id_j + \Delta$
  1. $(A_i, s_i, s'_i)_{i\in[m(\lambda)]} \leftarrow \mathsf{CosetGen}(1^\lambda, m(\lambda), \lambda; F_1(K_1, id))$.
  2. If $id < id_j + \Delta$, output $\perp$ and terminate.
  3. For each $i \in [m(\lambda)]$, check if $v_i \in A_i + s_i$ if $(x)_i = 0$ and check if $v_i \in A_i^\perp + s'_i$ if $(x)_i = 1$.
     If any of the checks fail, output $\perp$ and terminate.
  4. Output $f(x)$.
```

We now define some notation.

Definition 51. *Consider the following experiment.*

1. *Simulate the first two steps of $\mathcal{A}'_0$ and the challenger:*

   1.1. *Simulate $\mathcal{A}$ and the challenger. Let f be the function determined at the end of the setup phase and let $\mathsf{R}_{\mathsf{adv}}$ be the $(k+1)$-partite register (with state σ) output by $\mathcal{A}$ at the output phase.*

   1.2. *Uniformly at random sample x, y, j^* such that $1 \leq x < y \leq k+1$ and $j^* \in \{1, \ldots, k\}$.*

2. *Set $pp = (x, y, j^*, (id_j)_{j\in[k+1]}, f)$.*

3. *Output $\mathsf{R}_{\mathsf{adv}}, pp$.*

We will write $\mathsf{Exp}_{\mathcal{O}} \approx^c_\nu \mathsf{Exp}_{\mathcal{O}'}$ *to denote that the advantage of any QPT adversary in distinguishing the oracles* $\mathcal{O}, \mathcal{O}'$ *(which can depend on pp) given the outcome of the above experiment, is* ν*. We omit* ν *when* ν *is* $\mathsf{negl}(\lambda)$*.*

Claim 31. *We have*

- $\mathsf{Exp}_{\mathcal{O}_{j,\Delta}} \approx^c_{2^{-2\lambda}} \mathsf{Exp}_{\mathcal{O}_{j,\Delta+1}}$ *for all* $j \in \{0, 1, \dots, k\}$ *and* $\Delta \in \{1, \dots, id_{j+1} - id_j - 1\}$*,*
- $\mathsf{Exp}_{\mathcal{O}_{j,1}} \approx^c \mathsf{Exp}_{\mathcal{O}_{j+1}}$ *for all* $j \in \{0, 1, \dots, k\}$*,*
- $\mathsf{Exp}_{\mathcal{O}_0} \approx^c \mathsf{Exp}_{\mathcal{O}_1}$*.*

Proof. We prove the first point and the rest follow by the hybrid lemma through a simple calculation.

First, note that the oracles $\mathcal{O}_{j,\Delta}$ and $\mathcal{O}_{j,\Delta+1}$ only differ at points such that $id = id_j + \Delta$ and v_i are in the correct cosets (primal or dual) for the coset tuple generated using the randomness $F_1(K_1, id_j + \Delta)$. Let S denote the set of all such inputs and we claim $q_S \leq \mathsf{negl}(\lambda)$, that is, the adversary has negligible query weight on S. Suppose otherwise for a contradiction. Then, we can measure a random query of the adversary to obtain vectors as above with non-negligible probability. However, note the following:

- The adversary has only oracle access to the PRF key K, hence, $F_1(K_1, id_j + \Delta)$ is random given the adversary's view,
- The adversary never obtains an actual coset state for this tuple,
- The oracles $\mathcal{O}_f, \mathcal{O}_{j,\Delta}$ and $\mathcal{O}_{j,\Delta+1}$ can be simulated using a membership oracle for this tuple.

Since coset membership is unlearnable, by above we obtain a contradiction. Thus, $q_S \leq \mathsf{negl}(\lambda)$ and $\mathsf{Exp}_{\mathcal{O}_{j,\Delta}} \approx^c_{2^{-2\lambda}} \mathsf{Exp}_{\mathcal{O}_{j,\Delta+1}}$ by Theorem 11. □

Claim 32. *Let* τ *be the state of the bipartite register* $\mathsf{R}_{\mathsf{adv}}[x, y]$ *output by* $\mathcal{A}'_0$ *in* $\mathcal{G}$*, and also consider the classical values* $j^*, x, y, \{b_{\ell,i}\}_{\ell,i}$ *contained in the output of* $\mathcal{A}'_0$*.*

Suppose we apply the measurement $\mathsf{API}^{\varepsilon,\delta,\mathcal{O}_{j^*+1}}_{\mathcal{D}} \otimes \mathsf{API}^{\varepsilon,\delta,\mathcal{O}_{j^*+1}}_{\mathcal{D}}$ *to* τ *and let* b_{x,j^*+1}, b_{y,j^*+1} *denote the measurement outcomes we obtain. Then,*

$$\Pr\left[b_{x,j^*} - b_{x,j^*+1} > \frac{29\gamma}{32k} \wedge b_{y,j^*} - b_{y,j^*+1} > \frac{29\gamma}{32k}\right] > \frac{1}{\mathsf{poly}(\lambda)}$$

where the probability is taken over the randomness of the challenger, the adversary $\mathcal{A}'_0$ *and the measurement outcomes.*

Proof. First, note that

$$\Pr\left[\left(\mathsf{PI}^{\mathcal{O}_{k+1}}_{\mathcal{D}}\right) \cdot \iota \geq \frac{\gamma}{32}\right] = 0.$$

for any state ι that can be efficiently obtained during game $\mathcal{G}$, since $\mathcal{O}_{k+1}$ is the empty oracle and $(\mathcal{F}, \mathcal{D})$ is unlearnable. The rest follows from the same argument as in the proof Claim 5 and by the fact that $\mathsf{Exp}_{\mathcal{O}_0} \approx^c \mathsf{Exp}_{\mathcal{O}_1}$ (Claim 31). □

Claim 33. *Let* τ *be the bipartite state output by* $\mathcal{A}'_0$ *in* $\mathcal{G}$*. Let* p'_x, p'_y *be the outcome of applying* $\mathsf{PI}^{\mathcal{O}_{j^*}}_{\mathcal{D}} \otimes \mathsf{PI}^{\mathcal{O}_{j^*}}_{\mathcal{D}}$ *to* τ*. Similarly, let* p''_x, p''_y *be the outcome of applying* $\mathsf{PI}^{\mathcal{O}_{j^*,1}}_{\mathcal{D}} \otimes \mathsf{PI}^{\mathcal{O}_{j^*,1}}_{\mathcal{D}}$ *to* τ*. Then,*

- $\Pr\left[p'_x > b_{x,j^*} - \frac{3\gamma}{32k} \wedge p'_y > b_{y,j^*} - \frac{3\gamma}{32k}\right] \geq 1 - 2^{-3\lambda}$.

- $\Pr\left[b_{x,j^*} - p''_x > \frac{28\gamma}{32k} \wedge b_{y,j^*} - p''_y > \frac{28\gamma}{32k}\right] > \frac{1}{q(\lambda)}$ *for some polynomial* $q(\cdot)$.

Proof. Follows from the same arguments as in the proof of Claim 14, and by Claim 32. □

Now, we claim that we can extract correct vectors from the output state τ of the adversary. First, observe that we have $\mathsf{Exp}_{\mathcal{O}_{j^*}} \not\approx_c \mathsf{Exp}_{\mathcal{O}_{j^*1,1}}$, since $\mathsf{Exp}_{\mathcal{O}_{j^*}} \approx_c \mathsf{Exp}_{\mathcal{O}_{j^*1,1}}$ would give us a contradiction to Claim 33 by Theorem 14. Then, by the contrapositive of Theorem 11, we get that the freeloader encoded in $\tau[1]$ has a non-negligible query weight on the set of points of where the oracles $\mathcal{O}_{j^*}$ and $\mathcal{O}_{j^*,1}$ differ. Observe that these oracles only differ on points that satisfy $id = id_{j^*}$ and $v_i \in A_i^{j^*} + {s_i}^{j^*}$ if $(r_1)_i = 0$ and $v_i \in (A_i^{j^*})^\perp + {s'_i}^{j^*}$ if $(r_1)_i = 1$. Hence, by measuring a random query of the freeloader adversary encoded in $\tau[1]$ (which we simulate using a universal quantum circuit), we obtain correct coset vectors wtih non-negligible probability. We set this algorithm as our adversary $\mathcal{A}'_1$ for the game $\mathcal{G}$.

Finally, we claim that even conditioned a successful coset vector extraction from the first freeloader, we can still extract from the second freeloader. Let ξ denote the post-measurement state of the second register, obtained by extracting from the first register of τ as above using $\mathcal{A}'_1$ and conditioning on a successful extraction. We claim that ξ satisfies

1. $\Pr\left[\mathsf{PI}_{\mathcal{D}}^{\mathcal{O}_{j^*}} \cdot \xi \leq b_{y,j^*} - \frac{3\gamma}{32k}\right] \leq 2^{-\lambda}$.
2. $\Pr\left[\mathsf{PI}_{\mathcal{D}}^{\mathcal{O}_{j^*,1}} \cdot \xi < b_{y,j^*} - \frac{28\gamma}{32k}\right] \geq \frac{1}{\mathsf{poly}(\lambda)}$.

This first claim follows from Claim 33, Theorem 9, and the fact that extraction on the first register succeeds with non-negligible probability. To see the second point, observe that we can imagine $\mathsf{PI}_{\mathcal{D}}^{\mathcal{O}_{j^*,1}}$ being applied to the second register, before an extraction on the first register, and condition on obtaining a value $< b_{y,j^*} - \frac{28\gamma}{32k}$ (denote this as event G_2). Then, by Claim 33, it is easy to see that the first register still has a *gap* between $\mathcal{O}_{j^*}$ and $\mathcal{O}_{j^*,1}$, satisfying the properties we used to construct $\mathcal{A}'_1$ for an extraction from the first register. Let E_1 denote the event that $\mathcal{A}'_1$ succesfully extracts from the first register. Hence, by the foregoing discussion, we have $\Pr[E|G] > \frac{1}{\mathsf{poly}}$. We also have $\Pr[G] > \frac{1}{\mathsf{poly}}$ and $\Pr[E] > \frac{1}{\mathsf{poly}}$ from before. Thus, we get $\Pr[G|E] > \frac{1}{\mathsf{poly}}$, proving the second point (Item 2).

Given Item 1 and Item 2, by the same extraction argument we used for the first register, we conclude that there exists an adversary $\mathcal{A}'_2$ that extracts correct coset vectors from the second register of the output of $\mathcal{A}'_0$ with non-negligible probability conditioned on $\mathcal{A}'_1$ extracting correct vectors from the first register. Hence, we have that the adversary tuple $\mathcal{A}' = (\mathcal{A}'_0, \mathcal{A}'_1, \mathcal{A}'_2)$ wins the game $\mathcal{G}$ with non-negligible probability. It is easy to see that this gives us a contradiction by Theorem 24 (see Claim 16 for a similar reduction).

D Proof of Anti-Piracy Security of the PRF Scheme

In this section, we prove Theorem 41.

First, we show that hidden trigger inputs are indistinguishable from uniformly random challenge strings, even when the adversary gets a (obfuscated) program that allows it to generate its own hidden trigger inputs.

Definition 52 (Hidden Trigger Inputs)**.** *Let* $\mathsf{GenTrigger}_{K_0,K_3,K_4,\mathsf{OPMem},cpk}$ *be the following program, where the hardcoded values are as in the PRF scheme construction (Section 10.2). The input format to the program will be clear from context.*

```
$\mathsf{GenTrigger}_{K_0,K_3,K_4,\mathsf{OPMem},cpk}(r_1, r_2, r_3)$
    Hardcoded: $K_0, K_3, K_4, \mathsf{OPMem}, cpk$
    1. Parse $x_1||x_2||x_3 = G_2(r_1)$ with $|x_i| = s_i$.
    2. Parse $y||K_2' = G_1(F(K_0, x))$ with $|y| = n(\lambda)$.
    3. $\mathsf{OQ} \leftarrow i\mathcal{O}(Q_{cpk,\mathsf{OPMem},x_1,K_2',y}; r_3)$.
    4. $x_2' = F_3(K_3, x_1||\mathsf{OQ}||G_3(r_2))$.
    5. $x_3' = F_4(K_4, x_2') \oplus (x_1||\mathsf{OQ}||G_3(r_2))$.
    6. Output $x_1||x_2'||x_3'$.
```

The circuit $Q_{cpk,\mathsf{OPMem},x_1,K_2',y}$ used above is the following. Note that it contains hardcoded values that are computed during the execution of GenTrigger*.*

```
$Q_{cpk,\mathsf{OPMem},x_1,K_2',y}(w)$
    Hardcoded: $cpk, \mathsf{OPMem}, x_1, K_2', y$
    1. Parse $id, u_1, \dots, u_{c_L(\lambda)} = w$.
    2. Run $\mathsf{OPMem}(id, u_1, \dots, u_{c_L(\lambda)}, x_1)$. If it outputs $0$, output $\perp$ and terminate.
    3. Output $\mathsf{IBE.Enc}(cpk, id, y; F_2(K_2', id))$.
```

Lemma 22. *Consider the following game for the PRF scheme from Section 10.2, where we let $a(\lambda)$ denote the length of the randomness used by $i\mathcal{O}$ to obfuscate Q in Definition 52. Consider the following experiment, parameterized by $\ell(\lambda)$.*

$\mathsf{HiddenTriggerExp}(\lambda, \mathcal{A}, \ell(\lambda), b)$

1. *The challenger runs $K \leftarrow \mathsf{KeyGen}(1^\lambda)$.*
2. *For multiple rounds, $\mathcal{A}$ makes quantum key queries. For each query, the challenger generates a key as $\mathsf{R} \leftarrow \mathsf{QKeyGen}(K)$ and submits R to the adversary.*
3. *The adversary outputs a register $\mathsf{R}_{\mathsf{adv}}$.*
4. *Sample $\mathsf{OGenTrigger} \leftarrow i\mathcal{O}(\mathsf{GenTrigger})$.*
5. *For $i = 1$ to ℓ:*
    1. *Sample $r_1^i \leftarrow \{0,1\}^{s_1(\lambda)/2}$.*
    2. *Sample $r_2^i \leftarrow \{0,1\}^{\lambda}$.*
    3. *Sample $r_3^i \leftarrow \{0,1\}^{a(\lambda)}$.*
    4. *Set $z^{0,i} = \mathsf{OGenTrigger}(r_1^i, r_2^i, r_3^i)$.*
    5. *Sample $z^{1,i} \leftarrow \{0,1\}^{m(\lambda)}$.*
6. *Output $((z^{b,i})_{i\in[\ell]}, \mathsf{OGenTrigger}, \mathsf{R}_{\mathsf{adv}})$.*

Then, for any polynomial $\ell(\lambda)$,

$$\mathsf{HiddenTriggerExp}(\lambda, \mathcal{A}, \ell(\lambda), 0) \approx_c \mathsf{HiddenTriggerExp}(\lambda, \mathcal{A}, \ell(\lambda), 1).$$

Proof. Follows from Lemma 12. Note that the only difference is that the adversary no longer gets a verification program, which only makes the adversary weaker. □

We will prove anti-piracy security through a series of hybrids. Define Hyb_0 to be the original game $\mathsf{PRFAntiPiracy}(\lambda, \mathcal{A})$ from Definition 44.

$\underline{\mathsf{Hyb}_1}$: The challenger now computes the PRF challenge outputs $ch^{0,\ell}$ by sampling a new key using $\mathsf{QKeyGen}$ for each ℓ and using OPEval. By the correctness of the copy-protected PRF scheme, we will have $ch^{0,\ell} = F(K_0, x^\ell)$ for all $\ell \in [k+1]$ with overwhelming probability. Hence, $\mathsf{Hyb}_0 \approx \mathsf{Hyb}_1$.

$\underline{\mathsf{Hyb}_2}$: Instead of sampling $ch^{1,\ell} \leftarrow \{0,1\}^{m(\lambda)}$, we will now compute $ch^{1,\ell}$ by sampling a new key using $\mathsf{QKeyGen}$ for each ℓ and running OPEval on input q^ℓ, where we sample $q^\ell \leftarrow \{0,1\}^{m(\lambda)}$. With overwhelming probability, we will have $ch^{1,\ell} = F(K_0, q^\ell)$, and this is indistinguishable from a random string since q^ℓ is not given to the adversary and F is an extracting PRF. Hence, $\mathsf{Hyb}_1 \approx \mathsf{Hyb}_2$.

$\underline{\mathsf{Hyb}_3}$: We now sample x^ℓ, q^ℓ for all $\ell \in [k+1]$ as hidden triggers (Definition 52). We get $\mathsf{Hyb}_2 \approx \mathsf{Hyb}_3$ by Lemma 22. Crucially note that at the challenger no longer directly uses the PRF key K and instead computes the challenges using $\mathsf{QKeyGen}$ queries and OPEval, which is in adversary's view in Lemma 22. Hence, the adversary can indeed simulate $\mathsf{Hyb}_2, \mathsf{Hyb}_3$ in the reduction to Lemma 12. Hence, $\mathsf{Hyb}_2 \approx \mathsf{Hyb}_3$.

$\underline{\mathsf{Hyb}_4}$: We now sample x^ℓ, q^ℓ for $\ell \in [k+1]$ as follows.

1. Sample $r_1^\ell \leftarrow \{0,1\}^{s_1(\lambda)/2}$.
2. Sample $r_2^\ell \leftarrow \{0,1\}^\lambda$.
3. Sample $r_3^\ell \leftarrow \{0,1\}^{a(\lambda)}$.
4. Let $w^\ell = G_2(r_1^\ell)$.
5. Parse $w_1^\ell || w_2^\ell || w_3^\ell = w^\ell$ with $|w_i^\ell| = s_i$.
6. Parse $y^\ell || K_2^\ell = G_1(F(K_0, w^\ell))$ with $|y^\ell| = n(\lambda)$.
7. $\mathsf{OQ}^\ell \leftarrow i\mathcal{O}(Q_{cpk, \mathsf{OPMem}, w_1^\ell, K_2^\ell, y^\ell}; r_3^\ell)$.
8. $w_2'^\ell = F_3(K_3, w_1^\ell || \mathsf{OQ}^\ell || G_3(r_2^\ell))$.
9. $w_3'^\ell = F_4(K_4, w_2'^\ell) \oplus (w_1^\ell || \mathsf{OQ}^\ell || G_3(r_2^\ell))$.
10. Set $x^\ell = w_1^\ell || w_2'^\ell || w_3'^\ell$.
11. Sample $qr_1^\ell \leftarrow \{0,1\}^{s_1(\lambda)/2}$.
12. Sample $qr_2^\ell \leftarrow \{0,1\}^\lambda$.

13. Sample $qr_3^\ell \leftarrow \{0,1\}^{a(\lambda)}$.
14. Let $qw^\ell = G_2(qr_1^\ell)$.
15. Parse $qw_1^\ell||qw_2^\ell||qw_3^\ell = qw^\ell$ with $|qw_i^\ell| = s_i$.
16. Parse $qy^\ell||qK_2^\ell = G_1(F(K_0, qw^\ell))$ with $|qy^\ell| = n(\lambda)$.
17. $\mathsf{qOQ}^\ell \leftarrow i\mathcal{O}(Q_{cpk,\mathsf{OPMem},qw_1^\ell,qK_2^\ell,qy^\ell}; r_3^\ell)$.
18. $qw_2'^\ell = F_3(K_3, qw_1^\ell||\mathsf{qOQ}^\ell||G_3(qr_2^\ell))$.
19. $qw_3'^\ell = F_4(K_4, qw_2'^\ell) \oplus (qw_1^\ell||\mathsf{qOQ}^\ell||G_3(qr_2^\ell))$.
20. Set $q^\ell = qw_1^\ell||qw_2'^\ell||qw_3'^\ell$.

Note that this is only a syntactic change, we only unwrapped the sampling of the inputs x^ℓ, q^ℓ. Hence, $\mathsf{Hyb}_3 \equiv \mathsf{Hyb}_4$.

$\underline{\mathsf{Hyb}_5}$: We now sample w^ℓ, qw^ℓ for all $\ell \in [k+1]$ uniformly at random. We get $\mathsf{Hyb}_4 \approx \mathsf{Hyb}_5$ by the security of the PRG G_2.

$\underline{\mathsf{Hyb}_5}$: We now sample all y^ℓ, qy^ℓ and K_2^ℓ, qK_2^ℓ uniformly at random. First, observe that with overwhelming probability, the values w^ℓ, qw^ℓ for all $\ell \in [k+1]$ will be unique. Then, observe that the inputs w^ℓ, qw^ℓ to F have each min-entropy $s_2 + s_3$ given the view of the adversary, since only the first part w_1^ℓ, qw_1^ℓ is used during the hidden trigger generation and the rest is discarded. Therefore, by our choice of parameters for the extracting PRF F and by the security of PRG G_1, we have $\mathsf{Hyb}_5 \approx \mathsf{Hyb}_6$.

Finally, observe that the adversary gets either $(x^\ell, ch^{0,\ell} = (\mathsf{OQ}^\ell, w_1^\ell))$ (for $b_\ell = 0$) or $(x^\ell, ch^{1,\ell} = (\mathsf{qOQ}^\ell, qw_1^\ell))$ (for $b_\ell = 1$) since x^ℓ, q^ℓ are hidden trigger inputs and we obtain $ch^{0,\ell}, ch^{1,\ell}$ using OPEval. Observe that these are the same as ciphertexts of our PKE scheme (Section 7.2) encrypting the random messages y^ℓ and qy^ℓ respectively. Hence, the security follows by the CPA-style anti-piracy security (see Section 7.3) of our scheme and we have $\Pr[\mathsf{Hyb}_5 = 1] \leq 1/2 + \mathsf{negl}(\lambda)$.