Unclonable Cryptography with Unbounded Collusions

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

Vipul Goyal
NTT Research &
Carnegie Mellon University
vipul@cmu.edu

Abstract

Quantum no-cloning theorem gives rise to the intriguing possibility of quantum copy protection where we encode a program in a quantum state such that a user in possession of k such states cannot create $k+1$ working copies. Introduced by Aaronson (CCC'09) over a decade ago, copy protection has proven to be notoriously hard to achieve.

In this work, we construct public-key encryption and functional encryption schemes whose secret keys are copy-protected against *unbounded collusions* in the plain model (i.e. without any idealized oracles), assuming (post-quantum) subexponentially secure $i\mathcal{O}$, one-way functions and LWE. This resolves a long-standing open question of constructing fully collusion-resistant copy-protected functionalities raised by multiple previous works.

Prior to our work, copy-protected functionalities were known only in restricted collusion models where either an a-priori bound on the collusion size was needed, in the plain model with the same assumptions as ours (Liu, Liu, Qian, Zhandry [TCC'22]), or adversary was only prevented from doubling their number of working programs, in a structured quantum oracle model (Aaronson [CCC'09]).

We obtain our results through a novel technique which uses identity-based encryption to construct unbounded collusion resistant copy-protection schemes from $1 \to 2$ secure schemes. This is analogous to the technique of using digital signatures to construct full-fledged quantum money from single banknote schemes[1] (Lutomirski et al. [ICS'09], Farhi et al. [ITCS'12], Aaronson and Christiano [STOC'12]). We believe our technique is of independent interest.

Along the way, we also construct a 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.

[1] Also called *mini-schemes*

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*. In a quantum money scheme, 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 (polynomial) k cannot produce $k+1$ *authentic* banknotes. While the original scheme of [Wie83] required the involvement of the bank to verify a banknote, later work [AC12, Zha19] has succeeded in constructing what is called *publicly-verifiable* quantum money using *subspace states*, where anyone can verify the authenticity of a claimed banknote.

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 programs, where a user in possession of some number of copies of a program P cannot produce more 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 program P, you can make any number of copies of it. However, [Aar09] showed copy-protection using quantum information is indeed possible: in a structured quantum oracle model, any *unlearnable* program can be copy-protected in a way that is $k \to k+r$ secure (for any k and some $r > k$). By $a \to b$ copy-protection, we mean that any malicious user in possession of a instances of the program P cannot produce b working copies. For example, in the construction of [Aar09], the adversary is prevented from doubling their number of working copies. Later, Aaronson et al. [ALL+21] showed that in a classical structured oracle model any unlearnable program can be $1 \to 2$ copy-protected.

In a related line of work, Georgiou and Zhandry [GZ20] started the study of public-key encryption with copy-protected secret keys, also called *single-decryptor encryption*. In this model, a *pirate* adversary obtains the public key and k copy-protected secret keys of the scheme. Then, it attempts to produce $k+1$ *freeloader* adversaries that are later presented with challenge ciphertexts. We require that they cannot all succeed simultaneously. [GZ20] also gave a secure $1 \to 2$ copy-protected scheme in a structured oracle model. Later, Coladangelo et al. [CLLZ21] showed how to construct a $1 \to 2$ copy-protected scheme in the plain model using *coset states*, assuming quantum harndess of LWE, (post-quantum) subexponential indistinguishability obfuscation and one-way functions. Liu et al. [LLQZ22] showed through an elegant proof that the parallel repetition of the scheme of [CLLZ21] is $k \to k+1$ copy-protection secure. However, we need to know the collusion-bound k during setup, and the size of the scheme grows linearly with the bound k.

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?

By unbounded collusion-resistant copy-protection, we mean that we require $k \to k+1$ security for any (polynomial) k, where k may not be known at the time of setup and the size of the scheme does not depend on it. Aside from theoretical interest, note that such constructions are especially needed in various cases, such as functional encryption, where we could need to issue unbounded number of programs to the users.

In this work, we answer the above question positively, with assumptions matching the previous work.

1.1 Our Results

In this work, we resolve the open problem of constructing fully collusion-resistant copy-protection schemes by constructing such schemes for public-key encryption and public-key functional encryption. As a corollary, we also automatically obtain collusion-resistant unclonable constructions for primitives such as identity-based encryption, identity cards ([Aar09]) and attribute-based encryption.

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

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

We also achieve functional encryption with copy-protected quantum secret keys that are collusion-resistant against unbounded collusion, in the plain model.

Theorem 2. *Assuming post-quantum subexponentially secure indistinguishability obfuscation, one-way functions and 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 (Kitagawa and Nishimaki [KN22]) was in the $1 \to 2$ 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 1.2 for more detail). Furthermore, on top of matching the assumptions previous work used for constructing copy-protected public-key encryption, the assumptions 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].

An important contribution of our work is a novel technique to construct collusion-resistant copy-protection schemes which relies on using identity-based encryption. We use this technique in both 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 collusion-resistant quantum money from single banknote schemes [LAF+09, FGH+12, AC12].

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 after the challenge

[2] More specifically, our assumptions exactly match the assumptions made by [LLQZ22], but [CLLZ21] assumes polynomially secure LWE whereas we assume subexponentially secure LWE.

ciphertext in the usual functional encryption security game, since the adversary can issue their own keys using the punctured master secret key.

Theorem 3. *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 $f(m_0) \neq f(m_1)$.*

1.2 Related Work

Aaronson [Aar09] introduced the idea of copy-protecting programs using quantum states based on the no-cloning principle, and built copy-protection schemes for all unlearnable functions based on a structured quantum oracle. Their result also satisfies a notion of weak collusion-resistance where an adversary that obtains k copies of the program cannot produce $k + r$ copies for some $r > k$. Later, Aaronson et al. [ALL+21] showed how to build copy-protection schemes for all unlearnable functions using a classical oracle that depends on the program that is being copy-protected.

Georgiou and Zhandry [GZ20] started study of public-key encryption scheme with copy-protected quantum secret keys, which they term single-decryptor encryption, and gave a construction in a structured oracle model. Coladangelo et al. [CLLZ21] constructed a public-key encryption scheme with copy-protected quantum secret keys and a PRF scheme with copy-protected keys, both in the plain model using coset states. Later, Liu et al. [LLQZ22] showed that parallel repetition of the schemes of [CLLZ21] achieve $k \to k + 1$ and they also gave a signature scheme with copy-protected signing keys. However, for all of their constructions, the collusion bound k needs to be known during setup and the key and ciphertext sizes grow with k. Assumptions made by [LLQZ22] match ours exactly. The assumptions made by [CLLZ21] also match ours except that they assume polynomially secure LWE and their reductions have polynomial loss of security (though they still need subexponentially secure iO and one-way functions) - whereas we (and [LLQZ22]) need subexponential security of LWE and the reductions have subexponential loss of security. Kitagawa and Nishimaki [KN22] defined functional encryption with copy-protected functional keys in a weaker 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.

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 an honest algorithm. [ALP21] also show that even this weaker notion is impossible to achieve for all unlearnable programs, based on some standard assumptions. They also define a variant where we require that the adversary cannot produce a working copy and a valid deletion certificate at the same time. In another variation of the latter model defined for certain primitives, called certified key deletion or secure key leasing, we allow the adversary to use any algorithm to run their forged copies, rather than only the honest algorithm. Various work [ALP21, ALL+21, KNY21, KN22, BGG+23] construct secure software leasing schemes and secure certified key deletion schemes for various primitives such as functional encryption, PRFs, indistinguishability obfuscation, based on standard assumptions. [ALL+21] show how to construct secure leasing schemes for any watermarkable program using a watermarking scheme and a quantum money scheme.

Bitansky and Vaikuntanathan [BV18], Kitagawa2022 [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 schemes, 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 Technical Overview

2.1 Public-Key Encryption with Copy-Protected Secret Keys

Let us first describe our security model. In an anti-piracy game for public-key encryption, we have an adversary, called *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 $(m^0_\ell, m^1_\ell)_{\ell\in[k+1]}$ 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 $\mathsf{Enc}(pk, m^{b_\ell}_\ell)$. 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 $1/2 + \mathsf{negl}(\lambda)$. Note that 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}.

$1 \to 2$ Copy-Protection Secure Construction of Coladangelo et al. [CLLZ21]

As a warm-up, we will discuss the $1 \to 2$ copy-protection secure construction and its proof given by [CLLZ21] based on coset states, which also forms the base of our construction.

A coset state [CLLZ21] is a state of the form $\sum_{a\in A}(-1)^{\langle s',a\rangle}|a+s\rangle =: \left|A_{s,s'}\right\rangle$ 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. While it is easy to see that measuring this state in computational or Hadamard basis will destroy the information in the other one, our requirement is even more strict: Informally, we require that no efficient adversary can obtain vectors in both $A+s$ and $A^\perp + s'$ simultaneously, even with access to the membership oracles for $A+s, A^\perp+s'$. More formally, 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 $\left|A_{s,s'}\right\rangle$ and $i\mathcal{O}(A+s), i\mathcal{O}(A^\perp+s')$ to the adversary $\mathcal{A}_0$, where we overload the notation to let $A+s, A^\perp+s'$ also denote the membership checking programs in these cosets. Then, the adversary $\mathcal{A}_0$ outputs two (entangled) registers R_1, 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 an implicit variation used by [CLLZ21] and later formalized in a different context by [ÇGLZR23], we present $\mathcal{A}_0$ with multiple coset states (called *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 $A_i^\perp + s'_i$ depending on random challenge strings r_1, r_2 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 18). We call this variation multi-challenge version.

Now, we move onto the copy-protected public-key encryption construction of [CLLZ21]. During setup, we sample subspaces $A_i \subseteq \mathbb{F}_2^\lambda$ of dimension $\lambda/2$ and elements $s_i, s'_i \in \mathbb{F}_2^\lambda$. The tuple of coset states $\left|A_{i,s_i,s'_i}\right\rangle_{i\in[c(\lambda)]}$ becomes the quantum secret key, and we output $(i\mathcal{O}(A_i+s), i\mathcal{O}(A_i^\perp+s'_i))_{i\in c(\lambda)}$ as the public key. To encrypt a message m, we sample a random string r and an obfuscation $\mathsf{OP} \leftarrow i\mathcal{O}(\mathsf{PCt})$.

$\underline{\mathsf{PCt}(u_1, \ldots, u_{c(\lambda)})}$

Hardcoded: $m, r, (i\mathcal{O}(A_i + s), i\mathcal{O}(A_i^\perp + s'_i))_{i\in c(\lambda)}$

1. For each $i \in [c(\lambda)]$, 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 $\perp$ and terminate.
2. Output m.

Basically, PCt verifies if the given vectors are in the correct cosets with respect to r. 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.

We will now informally argue security. Suppose for now that we are using black-box obfuscation to compute OPCt instead of $i\mathcal{O}$. Suppose an adversary, given the coset state tuple and the public key, can create two freeloaders that are both capable of decrypting their challenge ciphertexts (OPCt_1, r_1) and (OPCt_2, r_2). By the security of black-box obfuscation, this means that these freeloaders query their obfuscated programs at vectors that are in the correct cosets with respect to r_1, r_2. This corresponds to exactly to multi-challenge version of MoE, hence, this adversary gives us a way of breaking the multi-challenge MoE game, which is a contradiction. Therefore, we conclude the security of the scheme.

Coladangelo et al. [CLLZ21] show that we do not need black-box obfuscation to prove the security and $i\mathcal{O}$ suffices. They show that using compute-and-compare obfuscation [WZ17], we can first replace the challenge ciphertexts with obfuscated compute-and-compare programs that verify the vectors $(u_i)_i$. Then, using the extractability guarantees of compute-and-compare obfuscation, we are again able to obtain vectors u_i that are in the correct cosets with respect to r, therefore reducing the security to multi-challenge MoE. One caveat is that to win the MoE game, we need to extract vectors from two entangled adversaries simultaneously, which is a challenge since once we extract from the first adversary, the state of the second adversary might collapse irreversibly to a useless state. However, [CLLZ21] shows that we can still achieve simultaneous extraction, using *projective implementations* [Zha20, ALL+21]. While we use the same technique in our proof, we defer this issue to the later sections and we refer the reader to [CLLZ21] for a warm-up on this technique.

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 $v \in A_i + s_i$ and $w \in A_i^\perp + s'_i$. 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 secret keys only satisfy $1 \to 2$ unclonability.

One natural solution is to try and employ quantum states that already possess a collusion-resistant unclonability guarantee, such as Haar random states, psuedorandom states or t-designs [LLQZ22]. This is indeed the approach employed by [Aar09] to achieve $k \to 2k$ copy-protection in a structured quantum oracle model. However, the problem is that there is no known way of employing such 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 an independent coset state tuple for each copy-protected secret key, and try to formalize the following pigeonhole-like fact: If an adversary that has obtained k copy-protected secret keys produces $k+1$ freeloaders, then two of these freeloaders must be using the same initial copy-protected secret key. Then, we can rely on the $1 \to 2$ copy-protection security of this independently sampled quantum key. This possess us with two challenges. First, since the adversary can do much more than just picking one of their keys and try to split this particular key into two, how do we show that this pigeonhole result still holds? It turns out that, as shown by the elegant proof of [LLQZ22], we can still prove the security of this scheme. The second problem is that, since we are sampling a new, independent coset state tuple for each copy-protected secret key, we need to also sample the corresponding obfuscated membership checking programs (which will allow us to create ciphertexts that can indeed by decrypted honestly by this secret key) and include it in the public-key. That is, our public-key now will be $(i\mathcal{O}(A_{i,j}+s_{i,j}), i\mathcal{O}(A^{\perp}_{i,j}+s'_{i,j}))_{i\in[k],j\in[c(\lambda)]}$ and our ciphertexts will roughly look like $(i\mathcal{O}(\mathsf{PCt}), r)$ where PCt is the following program.

$\underline{\mathsf{PCt}(i, u_1, \ldots, u_{c(\lambda)})}$

Hardcoded: $m, r, (i\mathcal{O}(A_{i,j}+s_{i,j}), i\mathcal{O}(A^{\perp}_{i,j}+s'_{i,j}))_{i\in[k],j\in[c(\lambda)]}$

1. For each $j \in [c(\lambda)]$, check if $u_j \in A_{i,j}+s_{i,j}$ if $(r)_j = 0$ and check if $u_j \in A^{\perp}_{i,j}+s'_{i,j}$ if $(r)_j = 1$. If any of the checks fail, output $\perp$ and terminate.
2. Output m.

Basically, our ciphertext now also takes in an index that specifies which coset state tuple (that is, which quantum secret key) we are using to decrypt, and verifies the given vectors with respect to this coset tuple. Since we are not allowed to enlarge the public-key any time a new copy-protected secret key is handed out, we need to know the number of independent coset tuples we will sample in advance. 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 grows with k, since the scheme consists of k independent instances of the $1 \to 2$ secure scheme of [CLLZ21].

Pseudorandom Coset States: A Collusion-Resistant Construction Using Black-Box Obfuscation

Before moving onto our solution, as a warm-up, we give a simplified solution using black-box obfuscation. As discussed above, the main challenge preventing the construction of [LLQZ22] from achieving unbounded collusion-resistance is the following fact: Since we are sampling an independent coset tuple for each copy-protected key, we need to include in the public-key the obfuscated membership checking programs $i\mathcal{O}(A_{i,j}+s_{i,j}), i\mathcal{O}(A^{\perp}_{i,j}+s'_{i,j})$ for this coset tuple, so that our ciphertexts (which are obfuscated programs) can verify this new coset state tuple. 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 challenge is the following. Instead of sampling truly random coset tuples, we sample them pseudorandomly. 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 $F(K, id)$. Observe that, for any polynomial number of quantum secret keys produced, they will all have unique identities with overwhelming probability. Then, by the security of the PRF scheme, to any efficient adversary, these states will be indistinguishable from truly random and independent coset state tuples. We will now be able to have a polynomial size public-key that allows us to verify any possible (honest) coset state tuple.

We now describe the rest of this construction in more detail in the (hopeful) model of black-box obfuscation.

- During setup, we sample a PRF key K. We compute OPMem as the (black-box) obfuscation of the following program PMem.

  $\underline{\mathsf{PMem}(id, u_1, \ldots, u_{c_L(\lambda)}, r)}$

  Hardcoded: K

  1. $(A_i, s_i, s'_i)_{i\in[c(\lambda)]} \leftarrow \mathsf{CosetGen}(1^\lambda; F(K, id))$.
  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.

  We output K as the classical secret key and OPMem as the public key.

- To sample a copy-protected quantum secret key, we sample a random identity $id \leftarrow \{0,1\}^\lambda$. Then, we sample a coset state tuple $\left|A_{i,s_i,s'_i}\right\rangle$ using the randomness $F(K, id)$. We output the coset state tuple and the identity id.

- To encrypt a message m, we sample random $r \leftarrow \{0,1\}^{c(\lambda)}$ and output $(i\mathcal{O}(\mathsf{PCt}), r)$ where PCt is the following program.

  $\underline{\mathsf{PCt}(id, u_1, \ldots, u_{c(\lambda)})}$

  Hardcoded: OPMem, r, m

  1. Run $\mathsf{OPMem}(id, u_1, \ldots, u_{c(\lambda)}, r)$. If it outputs 0, output $\perp$ and terminate.
  2. Output m.

- To decrypt a ciphertext, we apply QFT to the coset states at indices $(r)_i = 1$, and then run PCt coherently on id and the coset states.

We can (informally) argue security as follows. As discussed above, for any efficient adversary that obtains any (polynomial) number of quantum secret keys, they will all have unique identity strings with overwhelming probability. Then, by the security of black-box obfuscation, we can argue that the adversary has only query access to the PRF key K. Therefore, we can invoke the security of the PRF scheme and the fact that the identity strings are unique to show that the adversary's view is indistinguishable from having obtained k independent coset state tuples. While

we still have the problem of arguing that one cannot produce $k+1$ working keys from k coset state tuples, as discussed above, this has already been solved by [LLQZ22]. Therefore, we (informally) conclude the unbounded collusion-resistant copy-protection security of this scheme.

Removing the Black-Box Obfuscation

Now suppose that to compute the public-key, we use indistinguishability obfuscation of PMem rather than black-box obfuscation. 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 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 included in 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, it 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. This is because the adversary has access to K in some form, 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 want to make sure that the adversary can obtain the hidden message m only by running PCt on one of the identity strings that it has obtained from the challenger. To ensure this, we use the following approach, inspired by the classical delegatable functional encryption construction of Chandran et al. [CGJS15]. 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[3] 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 $id_1, \ldots, id_k$ - the only identities for which it has obtained the 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. 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! However, this is still too many, since in this case the scheme size[4] still grows with the collusion-bound k. To solve this issue, we can again employ the pigeonhole principle: Since the adversary has only k secret identity keys $sk_1, \ldots, sk_k$ of the IBE scheme, we can argue that two of the $k+1$ freeloaders must be using the same key sk_i, hence we only need to puncture K at id_i. To formalize this intuition, we will we use the techniques of [LLQZ22] and [CGJS15], however, we defer this discussion to later sections.

Finally, we present our new scheme in more detail.

- During setup, we sample a PRF key K. We create an IBE instance as $cpk, cmsk \leftarrow \mathsf{IBE.Setup}(1^\lambda)$. We compute $\mathsf{OPMem} \leftarrow i\mathcal{O}(\mathsf{PMem})$ where PMem is the program defined before. We output $K, cmsk$ as the classical secret key and OPMem, cpk as the public key.

[3] While the encryption is a randomized algorithm, we can still use it inside an obfuscated program by employing the standard trick of using a PRF to supply the randomness. See also [CGJS15].

[4] 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.

- To sample a copy-protected quantum secret key, we sample a random identity $id \leftarrow \{0,1\}^\lambda$. Then, we sample a coset state tuple $\left|A_{i,s_i,s'_i}\right\rangle$ using the randomness $F(K, id)$. We also sample the secret key $ck \leftarrow \mathsf{IBE.KeyGen}(cmsk, id)$. We output the coset state tuple, the identity id and the key ck.
- To encrypt a message m, we sample random $r \leftarrow \{0,1\}^{c(\lambda)}$ and a PRF key K', and we output $(i\mathcal{O}(\mathsf{PCt}), r)$ where PCt is the following program.

  $\mathsf{PCt}(id, u_1, \ldots, u_{c(\lambda)})$

  Hardcoded: OPMem, r, m, K'

  1. Run $\mathsf{OPMem}(id, u_1, \ldots, u_{c(\lambda)}, r)$. If it outputs 0, output $\perp$ and terminate.
  2. Output $\mathsf{IBE.Enc}(cpk, id, m; F'(K', id))$.

- To decrypt a ciphertext, we apply QFT to the coset states at indices $(r)_i = 1$, and then run PCt coherently on id and the coset states. Once we obtain the IBE ciphertext, we decrypt it using the key ck.

2.2 Public-Key Functional Encryption with Copy-Protected Secret Keys

In this section, we discuss our public-key functional encryption with copy-protected secret keys. Let us first describe our security model. In an anti-piracy game for public-key functional encryption, we have an adversary, called *pirate*. This adversary is given the public key pk, and then for any (polynomial) number of rounds, it makes queries for functions f. For each query, it also specifies a type, CLASSICAL or PROTECTED. If the type is the former, then the challenger sends a classical functional key for f to the adversary and also records this function f. If the type is the latter, the challenger sends a copy-protected quantum key for f to the adversary. After the query phase is done, the adversary outputs a pair of challenge messages (m^0, m^1) and $k+1$ (possibly entangled) *freeloader* adversaries, where k is the number of copy-protected keys it has queried. We require that $f(m^0) = f(m^1)$ for all functions f that were queried in CLASSICAL type. Then, the challenger samples challenge bits b_ℓ, and presents each freeloader with $\mathsf{Enc}(pk, m_\ell^{b_\ell})$. After receiving their challenge ciphertexts, the freeloaders make further functional key queries for multiple (polynomially many) rounds. We require that their queries satisfy $f(m^0) = f(m^1)$. At the end, 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 $1/2 + \mathsf{negl}(\lambda)$. See Section 7 for further discussion on our model.

Before our construction, let us first give a recap of the classical functional encryption construction of [CGJS15]. While they originally construct a delegatable functional encryption scheme from an hierarchical identity-based encryption scheme, we simplify it to usual functional encryption from usual identity-based encryption.

- During setup, we create an IBE instance $pk, msk \leftarrow \mathsf{IBE.Setup}(1^\lambda)$. We output pk, msk as the public- and the master secret key.
- To generate a functional key for a function described by a circuit f, we output $\mathsf{IBE.KeyGen}(msk, f)$, a secret key associated with the identity string f.
- To encrypt a message m, we sample a PRF key K', and we output $i\mathcal{O}(\mathsf{PCt})$ where PCt is the following program.

$\underline{\mathsf{PCt}(f)}$

Hardcoded: m, K'

1. Compute $a = f(m)$.
2. Output $\mathsf{IBE.Enc}(cpk, f, a; F'(K', id))$.

Intuitively, an adversary that has only obtained the keys for $f_1, \ldots, f_k$ will be only be allowed to decrypt $f_1(m), \ldots, f_k(m)$ by the security of the IBE scheme. However, to be able to rely on the security of IBE, we need to make sure that $F'(K', id)$ is indeed (pseudo-)random. Similar to our previous discussion, this poses a challenge since we are only using $i\mathcal{O}$ rather than black-box obfuscation. [CGJS15] solves this issue by employing subexponentially secure $i\mathcal{O}$ and by creating exponentially many hybrids of the challenge ciphertext indexed by t where we set $a = f(m^0)$ or $a = f(m^1)$ depending on the threshold $t < f$ that we gradually move. Then, to move between these hybrids, we only need to puncture the PRF key at a single point - t.

Now, we move onto our public-key functional encryption scheme with copy-protected secret keys, which can be seen as an amalgamation of our public-key encryption scheme and the technique above. We start with our public-key encryption scheme. Now, to generate a quantum secret key for a function f, we sample a random id as before, but now we generate the coset tuple using the randomness $id||f$. Basically, the coset states now become associated with both the function 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 encryption of $f(m)$ under the identity $id||f$. To argue security, using our previous technique, we will pin down two freeloaders that use the same identity string id, and hence the same coset state tuple, thus violating the monogamy-of-entanglement property.

Finally, we present our new scheme in more detail.

- During setup, we sample a PRF key K. We create an IBE instance as $cpk, cmsk \leftarrow \mathsf{IBE.Setup}(1^\lambda)$. We compute OPMem as before. We output $K, cmsk$ as the classical secret key and OPMem, cpk as the public key.

- To sample a copy-protected quantum secret key for a function f, we sample a random identity $id \leftarrow \{0,1\}^\lambda$. Then, we sample a coset state tuple $\left|A_{i,s_i,s_i'}\right\rangle$ using the randomness $F(K, id||f)$. We also sample the secret key $ck \leftarrow \mathsf{IBE.KeyGen}(cmsk, id||f)$. We output the coset state tuple, the identity $id||f$ and the key ck.

- To encrypt a message m, we sample random $r \leftarrow \{0,1\}^{c(\lambda)}$ and a PRF key K', and we output $(i\mathcal{O}(\mathsf{PCt}), r)$ where PCt is the following program.

$\underline{\mathsf{PCt}(id||f, u_1, \ldots, u_{c(\lambda)})}$

Hardcoded: OPMem, r, m, K'

1. Run $\mathsf{OPMem}(id||f, u_1, \ldots, u_{c(\lambda)}, r)$. If it outputs 0, output $\perp$ and terminate.
2. Output $\mathsf{IBE.Enc}(cpk, id, f(m); F'(K', id||f))$.

- To decrypt a ciphertext, we apply QFT to the coset states at indices $(r)_i = 1$, and then run PCt coherently on $id||f$ and the coset states. Once we obtain the IBE ciphertext, we decrypt it using the key ck.

Building and Using Puncturable Functional Encryption

As we will later discuss, in our copy-protection security proofs, we will crucially rely on a technique called *projective implementation* [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 technique will allow us to simultaneously extract vectors from two entangled freeloader adversaries. While projective implementations are in general inefficient, [Zha20] also give 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 (or, 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. 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 scheme to use an identity-based encryption scheme with puncturable master secret key (Section 5 and [CZDC19]). 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. As discussed before, 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 $pmsk$.

2.3 Proving Security

In this section, we give a high-level overview of the security proof of our public-key encryption construction. The security proof of our functional encryption construction follows similarly and we refer the reader to Section 7 for details. We also need identity-based encryption with puncturable master secret keys in our FE security proof. We refer the reader to Section 5 for details.

On a high level, our proof uses ideas from [LLQZ22] to extract MoE vectors from freeloaders through a pigeonhole argument, while relying on ideas from [CGJS15] to make the former's for-

malization of *the pigeonhole principle for freeloaders* still work in the face of exponentially many possible valid coset states.

An important tool in our (and previous works') proofs is the so-called projective implementation [Zha20]. The main use of projective implementations is that it allows us to argue that we can extract MoE vectors from entangled adversaries. Note that in general, applying the compute-and-compare obfuscation extractor on one of the adversaries might irreversibly damage the other one. Let us briefly discuss this technique. Let $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_0 = I - \mathcal{E}\}$ be a binary POVM. [Zha20] shows that there is a *projective* measurement indexed by a finite subset of $\mathbb{R}_{0\leq .\leq 1}$, denoted $\mathsf{PI}(\mathcal{E})$ such that the following procedure has the same distribution as $\mathcal{E}$ for any state ρ.[5]

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

In our anti-piracy game, we assume that the pirate adversary outputs the freeloaders as a quantum state describing a quantum circuit (with some hardwired state) that takes in a challenge ciphertext and outputs a prediction b'. The challenger executes the freeloader using an appropriate universal quantum circuit $\mathsf{U}_{quantum}$. Now, let $\mathcal{D}$ be a ciphertext distribution and let R_1, R_2 be a pair of registers containing the freeloaders output by the pirate adversary, and consider the following measurement.

1. Sample $b \leftarrow \{0,1\}$.
2. Sample $ct \leftarrow \mathcal{D}(m_i^b)$.
3. Sample $b' \leftarrow \mathsf{U}_{quantum}(R_i, ct_i)$.
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 corresponds to the final part of the anti-piracy game. Hence, we can modify our game so that instead of this measurement, we use its projective implementation and compare the result to $1/2 + \gamma(\lambda)$ for some inverse polynomial $\gamma(\lambda)$. [CLLZ21] proves that this game is indeed stronger than the original one, and we will prove security with respect to this game.[6]

The security argument of [CLLZ21] proceeds (informally) as follows. If we assume that an adversary wins the strong anti-piracy game, we get that $\Pr[(\mathsf{PI}_{\mathcal{D}} \otimes \mathsf{PI}_{\mathcal{D}}) \cdot (R_1, R_2) > (1/2 + \gamma, 1/2 + \gamma)]$ with non-negligible probability, where $\mathcal{D}$ is the honest ciphertext distribution. Using so-called canonical vectors of cosets, we can cast the coset vector verification part of the challenge ciphertext program PCt as a compute-and-compare program that releases the message m upon verification. Now, we can define a ciphertext distribution $\mathcal{D}'$ where PCt uses an obfuscated compute-and-compare program to verify the input vectors (u_i). It is easy to show that the adversary wins with respect to this distribution also, by correctness of the compute-and-compare obfuscation and the security of the outer obfuscation.

We can also define a ciphertext distribution $\mathcal{D}''$ where instead of the obfuscated compute-and-compare program, we use a simulated one (that we obtain using the obfuscation scheme) that does not actually contain the hidden message m. Then, it is easy to see that

$$\Pr[(\mathsf{PI}_{\mathcal{D}''} \otimes \mathsf{PI}_{\mathcal{D}''}) \cdot (R_1, R_2) > (1/2, 1/2)] = 0$$

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

[6] 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.

since the challenge ciphertext is independent of the challenge bit b. Therefore, we get that there is a *jump* between the two measurements $\mathsf{PI}_{\mathcal{D}'}$ and $\mathsf{PI}_{\mathcal{D}''}$. Using the extractability guarantee of the compute-and-compare obfuscation scheme, this means that there is an efficient procedure that can extract MoE vectors using R_1. The cruicial point is that, since PI is a projective measurement, one can show that no matter which channel we apply to R_1, the second register R_2 still satisfies this gap between $\mathcal{D}'$ and $\mathcal{D}''$. This allows us to argue that we can again use the extractor given by the compute-and-compare obfuscation, hence simultaneously obtaining MoE vectors from both adversaries. This allows us to break the MoE game (which is a contradiction).

Now, we move onto the $k \to k+1$ unclonability proof of [LLQZ22]. Again, let us assume that a pirate adversary can win the anti-piracy game. In this case, our ciphertexts will be of the form $(ct_1, \ldots, ct_k)$, where each ct_i is an encryption obtained using an instance of the $1 \to 2$ secure scheme. We define a sequence of distributions $\mathcal{D}_0, \ldots \mathcal{D}_k$, where $\mathcal{D}_i$ is the ciphertext distribution with the first i (mini) ciphertexts being simulated ones that do not contain the message and the rest being honestly sampled ones. In this case, we know that for each freeloader $\ell \in [k]$, there must be an index i_ℓ where a *jump* happens between $\mathcal{D}_{i_\ell}$ and $\mathcal{D}_{i_\ell+1}$, since $\Pr[\mathsf{PI}_{\mathcal{D}_0} R_i > 1/2 + \gamma]$ is non-negligible but $\Pr[\mathsf{PI}_{\mathcal{D}_k} R_i > 1/2] = 0$. This informally means that the ℓ-th freeloader is using the ciphertext ct_{i_ℓ}. Crucially note that there are $k+1$ freeloaders while each $i_\ell \in \{0, 1, \ldots, k-1\}$. Hence, by pigeonhole principle, there must be two freeloaders $\ell, \ell' \in [k+1]$ such that $i^* = i_{\ell'} = i_\ell$, meaning that they are both using the i^*-th key to decrypt. Since measuring might irreversibly destroy the state of the freeloaders, we cannot actually test and find these indices ℓ, ℓ', i^*. However, as argued by [LLQZ22], we can simply randomly guess them and we only incur a polynomial loss. Finally, if we are right in our guess, we will be able to violate the $1 \to 2$ security of the i^*-th key.

Finally, we move onto a sketch of the security proof of our scheme. Similar to above, we can define ciphertext distributions $\mathcal{D}_j$, but for all $j \in \{0, 1, \ldots, 2^\lambda\}$, representing all possible identity strings in $\{0,1\}^\lambda$ (plus, the dummy upper 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 following program.

$\mathsf{PCt}^j(id, u_1, \ldots, u_{c(\lambda)})$

Hardcoded: OPMem, r, m, K'

1. Run $\mathsf{OPMem}(id, u_1, \ldots, u_{c(\lambda)}, r)$. If it outputs 0, output $\perp$ and terminate.
2. If $id < j$, set $a = \top$. Otherwise, we set $a = m$.
3. Output $\mathsf{IBE.Enc}(cpk, id, a; F'(K', id))$.

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. Similar to above, we have that $\Pr[\mathsf{PI}_{\mathcal{D}_0} R_i > 1/2 + \gamma]$ is non-negligible but $\Pr\left[\mathsf{PI}_{\mathcal{D}_{2^\lambda}} R_i > 1/2\right] = 0$. However, the problem is that now the jump points i_ℓ are in $\{0, 1, \ldots, 2^\lambda - 1\}$, whereas we only have $k+1$ freeloaders. Hence, we cannot apply the pigeonhole principle to guarantee that there is a pair of freeloaders ℓ, ℓ' that have $i_\ell = i_{\ell'}$. However, a careful reader might guess that the distributions above actually *collapse* around k points: $j = id_1, \ldots, id_k$, the identity strings of the secret keys obtained by the adversary. More formally, we claim that jumps can only happen at indices j that correspond to some id_q. The reason is that, the difference between $\mathcal{D}_j$ and $\mathcal{D}_{j+1}$ only occurs when the obfuscated program PCt 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_q, then the different outputs of these programs will be IBE ciphertexts that

are indistinguishable to the adversary. Therefore, no freeloader can detect this change, and there cannot be a jump between $\mathsf{PI}_{\mathcal{D}_j} R_i$ and $\mathsf{PI}_{\mathcal{D}_{j+1}} R_i$. This allows us to conclude that all jump points i_ℓ will be in $\{id_1, \ldots, id_k\}$, hence, we can again apply a pigeonhole argument as above. The full proof delicately intertwines all these observations, whilst also ensuring that we are using efficient approximate projective implementations in our reduction rather than inefficient PI. We refer the reader to Section 6.3 for details.

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 $\mathrm{poly}(\cdot)$ 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 $\|X - Y\|_{Tr} < \varepsilon$; and we write $X \approx^c_\varepsilon Y$ to mean $\left|\Pr\left[\mathcal{A}(1^\lambda, X) = 1\right] - \Pr\left[\mathcal{A}(1^\lambda, Y) = 1\right]\right| < \varepsilon$ 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.

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.

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* $\{F : \{0,1\}^{k(\lambda)} \times \{0,1\}^{m(\lambda)} \to \{0,1\}^{n(\lambda)}\}_{\lambda \in \mathbb{N}^+}$ *with the following efficient algorithms.*

- $F.\mathsf{Setup}(1^\lambda)$: *Takes in a security parameter and outputs a key in* $\{0,1\}^{k(\lambda)}$.

- $F(K, x)$:[7] *Takes in a key and an input, outputs an evaluation of the PRF.*
- $F.\mathsf{Puncture}(K, x)$: *Takes as input a key and a set $S \subseteq \{0,1\}^{m(\lambda)}$, outputs a punctured key.*

We require that 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 $s \in S$.*
4. *The adversary outputs a guess b' and we say that the adversary has won if $b' = b$.*

Theorem 4 ([SW14, GGM86, Zha12a])**.** *If (post-quantum) one-way functions exist, then for any efficiently computable functions $n(\cdot), m(\cdot)$, there exists a (post-quantum) puncturable PRF with input space $\{0,1\}^{n(\lambda)}$ and output space $\{0,1\}^{m(\lambda)}$.*

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[8] puncturable PRF against subexponential time adversaries with input space $\{0,1\}^{n(\lambda)}$ and output space $\{0,1\}^{m(\lambda)}$.

3.3 Indistinguishability Obfuscation

In this section, we introduce indistinguishability obfuscation.

Definition 2. *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, $\Pr\left[\tilde{C}(x) = C(x) : \tilde{C} \leftarrow i\mathcal{O}(1^\lambda, C)\right] = 1$.*

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

[8] 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.

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 $\Pr\left[\forall x\ C_0(x) = C_1(x) : (C_0, C_1, R_{\mathsf{aux}}) \leftarrow \mathcal{B}(1^\lambda)\right] \geq 1 - \mathsf{negl}(\lambda)$. Then, for any QPT adversary $\mathcal{A}$,

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

3.4 Functional Encryption

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

Definition 3 (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 5 (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 (measurement statistics) 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. □

Theorem 6. *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 $p_i = \mathrm{Tr}\{(M_i \otimes I)\rho\}$. Then,

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

Proof. See Appendix A.2. □

Theorem 7 (Quantum Union Bound for Commuting Projectors)**.** *Let $\Pi_1, \ldots, \Pi_n$ be a set of commuting projectors. Then, for any state ρ of appropriate dimension,*

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

Proof. While this is a folklore result, we give a proof in Appendix A.3 for completeness. □

3.6 Compute-and-Compare Obfuscation

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

Definition 4 (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 is sub-exponentially unpredictable if for any QPT adversary, given the auxiliary information R_{aux} and the description of f, the adversary can predict the target value y with at most subexponential probability.

Definition 5. *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 $(\mathsf{CCObf.Obf}(f, y, z), R_{\mathsf{aux}}) \approx (\mathsf{CCObf.Sim}(1^\lambda, |f|, |y|, |z|), R_{\mathsf{aux}})$ *where* $(f, y, z), R_{\mathsf{aux}} \leftarrow \mathcal{D}(1^\lambda)$.

Theorem 8 ([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.[9]

[9] 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.

3.7 Projective and Threshold Implementations

Definition 6 (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 7 ((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 9 (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.

Theorem 10 (Approximate Projective Implementation [Zha20]). *Let $\mathcal{P} = \mathcal{P} = \{P_i, I - P_i\}_{i \in \mathcal{I}}$ 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 $0 < \varepsilon, \delta \leq 1$, 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.*
- *$\Delta^{\varepsilon}_{\mathsf{Shift}}(\mathsf{API}^{\varepsilon,\delta}_{\mathcal{P},\mathcal{D}}, \mathsf{PI}(\mathcal{P}_{\mathcal{D}})) \leq \delta$.*
- *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 11 ([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,*

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

We also give the following generalization where we consider measurements over multiple registers and allow the measured state and the measurement to be correlated, which will be needed in our copy-protection proofs.

Theorem 12. *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 $(\mathcal{S}^0, \mathcal{B}^0_\ell) \approx^{\mathrm{c}}_{\nu(\lambda)} (\mathcal{S}^1, \mathcal{B}^1_\ell)$ for all $(\frac{k(\lambda)}{\mu^2(\lambda)} \cdot \mathrm{poly}(\lambda))$-time adversaries for each $\ell \in [k]$ for some ν, μ satisfying $\nu(\lambda) < \mu^2(\lambda)\mathrm{poly}(\lambda)$. Then,*

$$|\vec{p}_0 - \vec{p}_1| \leq \mu(\lambda).$$

Proof. See Appendix A.4. □

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

Theorem 13 (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* $\geq \eta$*.*

The above can also be generalized to multipartite systems as follows.

Theorem 14. *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.5 □

Theorem 15. *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,*

$$\Pr\left[\forall\ell\in[k]\quad(\vec{p'})_\ell\leq(\vec{p})_\ell+2\varepsilon\right]\geq1-2\cdot k\cdot\delta$$
$$\Pr\left[\forall\ell\in[k]\quad(\vec{p'})_\ell\geq(\vec{p})_\ell-2\varepsilon\right]\geq1-2\cdot k\cdot\delta.$$

Proof. See Appendix A.6. □

Theorem 16. *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,*

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

Proof. See Appendix A.7. □

4 Coset States

In this section, we start by giving the definition of coset states [CLLZ21] and 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 8 (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 $\left|A_{s,s'}\right\rangle$, to be*

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

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.

Fact 1 ([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 17 (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* R_1, R_2.
5. *For* $j\in\{1,2\}$, *run* $v_j \leftarrow \mathcal{A}_j(R_j, A)$.
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* $\mathcal{A}$,

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

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*

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

for all sufficiently large λ.

In the previous constructions of unclonable primitives [CLLZ21, LLQZ22], and also in our constructions, we will require the *freeloader* adversaries to output a vector from either $A+s$ or $A^\perp+s'$, depending on a random bit presented to them. However, the *pirate* adversary can always guess the challenge bit and measure the coset 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 [ÇGLZR23]. For completeness, we state and prove our version in full detail.

Definition 9. *Define* $\mathsf{CosetGen}$ *to be the following algorithm, where we set* $\kappa(\lambda)=\lambda^{\lceil 3/C_{\mathsf{MoE}}\rceil}$.

$\underline{\mathsf{CosetGen}(1^\lambda)}$

1. *For* $i \in [3\cdot\lambda^3]$*, 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* $(A_i, s_i, s'_i)_{i\in[3\cdot\lambda^3]}$.

We call the output of CosetGen *a coset tuple.*

Theorem 18 (Monogamy-of-Entanglement Property for Coset States - Multi-Challenge Version)**.** *Consider the following game between the challenger and an adversary tuple* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$.

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

1. $(A_i, s_i, s'_i)_{i\in[3\cdot\lambda^3]} \leftarrow \mathsf{CosetGen}(1^\lambda)$.
2. *For* $i \in [3\cdot\lambda^3]$,
   - 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* $\left\{\left|A_{i,s_i,s'_i}\right\rangle\right\}_{i\in[3\cdot\lambda^3]}, (\mathsf{OP}_i^0, \mathsf{OP}_i^1)_{i\in[3\cdot\lambda^3]}$ *to* $\mathcal{A}_0$.
4. $\mathcal{A}$ *outputs two (possibly entangled) registers* R_1, R_2.
5. *Challenger samples* $r_1 \leftarrow \{0,1\}^{3\cdot\lambda^3}$ *and* $r_2 \leftarrow \{0,1\}^{3\cdot\lambda^3}$.
6. *For* $\ell \in \{1,2\}$*, run* $(v_{\ell,i})_{i\in[3\cdot\lambda^3]} \leftarrow \mathcal{A}_j(R_j, r_j, (A_i)_{i\in[3\cdot\lambda^3]})$.
7. *For* $\ell \in \{1,2\}$ *and all* $i \in [3\cdot\lambda^3]$*, 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, 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, then there exists a constant* $C_{\mathsf{MoE.MultChal}} > 2$ *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.

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, $\Pr[\mathsf{MoE}-\mathsf{MultChal}(\lambda, \mathcal{A}) = 1] \geq \varepsilon(\lambda)$. Define Hyb_0 to be $\mathsf{MoE}-\mathsf{MultChal}(\lambda, \mathcal{A})$ and define Hyb_1 by modifying it as follows. The challenger uniformly samples r_1, r_2 so that there is an index $i \in [3\cdot\lambda^3]$ satisfying $(r_1)_i = 0$ and $(r_2)_i = 1$. Since sampling r_1, r_2 uniformly and independently satisfies this with probability $1 - (3/4)^{3\cdot\lambda^3}$, the distance between the two distributions is $(3/4)^{3\cdot\lambda^3}$, hence we get that $\mathcal{A}$ wins in Hyb_1 with probability at least $\varepsilon(\lambda) - (3/4)^{3\cdot\lambda^3}$.

For any $j^* \in [3\cdot\lambda^3]$, we define the adversary $\mathcal{A}^{j^*} = (\mathcal{A}_0^{j^*}, \mathcal{A}'_1, \mathcal{A}'_2)$ for MoE as follows.

$\underline{\mathcal{A}_0^{j^*}}$

On input a state ρ and the obfuscated programs $\mathsf{OP}^{0*}, \mathsf{OP}^{1*}$, sample r_1, r_2 so that there is an index $i \in [3 \cdot \lambda^3]$ such that $(r_1)_i = 0$ and $(r_2)_i = 1$. Set $\rho_{j^*} = \rho$, $\mathsf{OP}^0_{j^*} = \mathsf{OP}^{0*}$ and $\mathsf{OP}^1_{j^*} = \mathsf{OP}^{1*}$. For all $j \in [3 \cdot \lambda^3] \setminus \{j^*\}$, 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 $\mathcal{A}((\rho_j, \mathsf{OP}^0_j, \mathsf{OP}^1_j)_{j\in[3\cdot\lambda^3]})$ to obtain a bipartite state σ. Finally, output

$$((\sigma[1], (A_j)_{j\in[3\cdot\lambda^3]\setminus\{j^*\}}, j^*, r_1), (\sigma[2], (A_j)_{j\in[3\cdot\lambda^3]\setminus\{j^*\}}, j^*, r_2)).$$

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

$\mathcal{A}'_\ell$ runs $\mathcal{A}_\ell$ on its own input and the subspace description A it obtains from the challenger. 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 $\mathcal{A}'_\ell$.

Observe that for any fixed value of j^*, the input to $\mathcal{A}$ is distributed the same as the output of $\mathsf{CosetGen}$. Therefore, $\mathcal{A}^{j*}$ above obtains correct vectors (i.e., in $v \in A_j + s_j$ or $A_j^\perp + s'_j$ depending on $(r_\ell)_j$ for all $j \in [3 \cdot \lambda^3]$ and $\ell \in \{1,2\}$) with probability at least $\varepsilon(\lambda) - (3/4)^{3\cdot\lambda^3}$, since it perfectly simulates $\mathcal{A}$ playing Hyb_1.

Finally, we define an adversary $\mathcal{A}' = (\mathcal{A}'_0, \mathcal{A}''_1, \mathcal{A}''_2)$ for MoE where $\mathcal{A}'_0$ samples j^* uniformly at random, and then simulates $\mathcal{A}_0^{j^*}$. We similarly define $\mathcal{A}''_1$ and $\mathcal{A}''_2$. By above, $\mathcal{A}'$ obtains the correct vectors with probability $\varepsilon(\lambda) - (3/4)^{3\cdot\lambda^3}$. Moreover, with probability $1/(3 \cdot \lambda^3)$, independent of $\mathcal{A}'$ obtaining the correct vectors, j^* will be such that $(r_1)_{j^*} = 0$ and $(r_2)_{j^*} = 1$. Note that when j^* satisfies this and the vectors obtained by $\mathcal{A}'$ are correct, $\mathcal{A}'$ wins MoE. Hence, $\mathcal{A}'$ wins with probability at least $\frac{\varepsilon(\lambda)-(3/4)^{3\cdot\lambda^3}}{3\cdot\lambda^3}$. This completes the proof by our choice of parameters.

□

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 some 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 19 (Monogamy-of-Entanglement Property for Coset States - Collusion-Resistant Version)**.** *Let $L(\lambda)$ be a polynomial, denoting the length of the identity strings. Define $c_L(\lambda) = 3 \cdot (L(\lambda) + \lambda)^3$. 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* $\mathsf{OPMem} \leftarrow i\mathcal{O}(\mathsf{PMem}_K, 1^\lambda)$*, 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. ***Split:*** *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. *The challenger samples* $r_1, r_2 \leftarrow \{0,1\}^{c_L(\lambda)}$.
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* $L(\lambda)$ *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, then there exists a constant* $C_{\mathsf{MoE.Coll}} > 0$ *such that for any polynomial* $L(\lambda)$ *and for any QPT adversary tuple*

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

for all sufficiently large λ.

Proof. We will only prove security for the adaptive case, and the selective security case also follows similarly. Our proof will rely on *complexity leveraging*, i.e. basically guessing the challenge identity id^*.

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 4).

We first define a stronger game as follows.

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 $\mathsf{OPMem} \leftarrow i\mathcal{O}(\mathsf{PMem}_K, 1^\lambda)$, where PMem_K is the following program.

   $\mathsf{PMem}_K(id, u_1, \ldots, u_{c_L(\lambda)}, r)$

   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 adversary 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 $r_1, r_2 \leftarrow \{0,1\}^{c_L(\lambda)}$.
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 $v_{\ell,i} \in (A^*)^\perp_i + s'^*_i$ 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 strong 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 $(\bot, \bot)$.

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

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 $\mathsf{MoE-MultChal}$ with probability $2^{-2\cdot(\lambda+L(\lambda))}$, which is a contradiction by Theorem 18. 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 $i \in [c_L(\lambda)]$, $\mathsf{OP}^{*0}_i \leftarrow i\mathcal{O}(A^*_i + s^*_i)$ and $\mathsf{OP}^{*1}_i \leftarrow i\mathcal{O}(A^{*\bot}_i + s'^*_i)$ (i.e., as in Theorem 18) using $(A^*_i, s^*_i, s'^*_i)_{i\in c_L(\lambda)}$. Then, it samples $\mathsf{OPMem} \leftarrow i\mathcal{O}(\mathsf{PMem}'_{K\{id'\}, id', (\mathsf{OP}^{*0}_i, \mathsf{OP}^{*1}_i)_{i\in[c_L(\lambda)]}})$.

$\mathsf{PMem}'_{K\{id'\}, id', (\mathsf{OP}^{*0}_i, \mathsf{OP}^{*1}_i)_{i\in[c_L(\lambda)]}}(id, u_1, \ldots, u_{c_L(\lambda)}, r)$

Hardcoded: $K\{id'\}, id', (\mathsf{OP}^{*0}_i, \mathsf{OP}^{*1}_i)_{i\in[c_L(\lambda)]}$

1. If $id = id'$, execute the following.
   1.1. For each $i \in [c_L(\lambda)]$, check if $\mathsf{OP}^{*0}_i(u_i) = 1$ if $(r)_i = 0$ and check if $\mathsf{OP}^{*1}_i(u_i) = 1$ 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 $u_i \in A^\bot_i + s'_i$ 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 $\mathsf{OP}^{*0}_i, \mathsf{OP}^{*1}_i$, and by security of the obfuscation of 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 $K\{id'\}$[10]. 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.

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 18). □

5 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 at all functions such that $f(m_0) \neq f(m_1)$. Then, we show how to construct such schemes in the plain model.

5.1 Definitions

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

Definition 10. *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*

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

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

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

Definition 11 (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 6.2), 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 12. *Identity-based encryption with puncturable master secret key is an identity-based encryption scheme (Definition 10) 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 13 (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 14 (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 20. *Let* IBE *be an identity-based encryption scheme that satisfies [polynomial, subexponential] puncturable master secret key security (Definition 14). Then, it also satisfies [polynomial, subexponential] selective indistinguishability-based security (Definition 10).*

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 $\varepsilon(\lambda) = 1/\mathrm{poly}(\lambda)$ 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 1.2. However, throughout the paper, we will write *puncturable functional encryption* to mean our definition.

Definition 15 (Puncturable Functional Encryption)**.** *Puncturable functional encryption is a functional encryption scheme (Definition 3) 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 16 (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 17 (Puncturable Functional Encryption Security)**.** *Consider the following game between the challenger and an adversary $\mathcal{A}$.*

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

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 $\mathcal{A}$,

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

5.2 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.

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

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

$\underline{\mathsf{IBE.KeyGen}(msk, id)}$

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

$\underline{\mathsf{IBE.Punc}(msk, id)}$

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

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

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

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

Theorem 21. IBE *satisfies both correctness (Definition 13) and strong punctured master secret key correctness (Definition 14).*

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 22. IBE *satisfies puncturable master secret key security (Definition 14).*

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 4), we get the following corollary.

Corollary 1. *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 22 in Section 5.3. It is easy to see that our proof also generalizes to the subexponential security case. Hence, we get the following.

Corollary 2. *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.*

5.3 Proof of Security for Puncturable IBE

In this section, we prove Theorem 22. 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 14) 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.

5.4 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 7), 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 14) and deterministic identity key generation satisfying strong punctured key correctness (Definition 13)
- 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 $pk, (imsk, \mathsf{FULL-KEY})$.

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

1. Parse $(msk'', \mathsf{TYPE}) = msk'$.
2. If $\mathsf{TYPE} = \mathsf{PUNC} - \mathsf{KEY}$, 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 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 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 23. FE *satisfies both correctness (Definition 16) and strong punctured master secret key correctness (Definition 17).*

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

Theorem 24. FE *satisfies puncturable master secret key security (Definition 17).*

Since we construct in Section 5.2 a puncturable IBE with the properties required by FE based on $i\mathcal{O}$ and one-way functions, we get the following corollary.

Corollary 3. *Assuming the existence of subexponentially secure indistinguishability obfuscation and one-way functions, there exist a puncturable functional encryption scheme.*

5.5 Proof of Security for Puncturable Functional Encryption

In this section, we prove Theorem 24. 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 $\{0, 1, \ldots, 2^Q\}$.

Suppose for a contradiction there exists a QPT adversary $\mathcal{A}$ that wins the puncturable functional encryption game with non-negligible advantage, that is, $\Pr[\mathsf{PUN-FE-IND}(\lambda, \mathcal{A}) = 1] \geq 1/2 + 1/p(\lambda)$ 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 $\mathsf{PUN-FE-IND}(\lambda, \mathcal{A})$.

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

$\underline{\mathsf{PCt}^{(t)}(f)}$

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

Proof. Observe the programs $\mathsf{PCt}^{(t)}$ and $\mathsf{PCt}^{(t+1)}$ 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** $t \in \{0,1,\ldots,2^Q\}$**:** 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.

$\mathsf{Hyb}_{t,2}$ **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.

$\mathsf{Hyb}_{t,3}$ **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 $\mathsf{OPKey} \leftarrow \mathsf{PKey}^{(t)}$.

$\mathsf{PKey}^{(t)}(f)$

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).

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

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

$\mathsf{Hyb}_{t,6}$ **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 4. $|(\mathsf{Hyb}_t)_{|\overline{E_t}} - (\mathsf{Hyb}_{t,1})_{|\overline{E_t}}| < 2^{-\lambda - Q(\lambda)}$ *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 5. $|(\mathsf{Hyb}_{t,1})_{|\overline{E_t}} - (\mathsf{Hyb}_{t,2})_{|\overline{E_t}}| < 2^{-\lambda - Q(\lambda)}$ *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 6. $|(\mathsf{Hyb}_{t,2})_{|\overline{E_t}} - (\mathsf{Hyb}_{t,3})_{|\overline{E_t}}| < 2^{-\lambda - Q(\lambda)} \cdot \mathrm{poly}(\lambda)$ *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 7. $|(\mathsf{Hyb}_{t,3})_{|\overline{E_t}} - (\mathsf{Hyb}_{t,4})_{|\overline{E_t}}| < 2^{-\lambda - Q(\lambda)}$ *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 14). □

Lemma 8. $|(\mathsf{Hyb}_{t,4})_{|\overline{E_t}} - (\mathsf{Hyb}_{t,5})_{|\overline{E_t}}| < 2^{-\lambda - Q(\lambda)} \cdot \mathrm{poly}(\lambda)$ *for all* $t \in \{0, 1, \ldots, 2^Q\}$.

Proof. Same argument as Lemma 6. □

Lemma 9. $|(\mathsf{Hyb}_{t,5})_{|\overline{E_t}} - (\mathsf{Hyb}_{t,6})_{|\overline{E_t}}| < 2^{-\lambda - Q(\lambda)}$ *for all* $t \in \{0, 1, \ldots, 2^Q\}$.

Proof. Same argument as Lemma 5. □

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

Proof. Same argument as Lemma 4. □

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).

6 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.

6.1 Definitions

Definition 18 (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*.*
- $\mathsf{Dec}(R_{\mathsf{dec}}, ct)$*: Takes in a quantum secret key and a ciphertext, outputs a message or* $\perp$*.*

We require correctness and CPA security*.*

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

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

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 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 definitions follow [CLLZ21, LLQZ22], with the strengthening that we allow unbounded[11] number of key queries and we also allow the adversary to choose different challenge messages for each freeloader.

Below, we write $\mathsf{U}_{quantum}$ to denote the quantum universal circuit $\mathsf{U}_{quantum}((\mathcal{B}, \rho), x)$ that takes in the description of a quantum circuit $\mathcal{B}$ with a hardwired state ρ and simulates it on input x. That is, it computes $\mathcal{B}(\rho, x)$.

Definition 19 (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 $\mathcal{A}$.*

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

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 $R \leftarrow \mathsf{PKE.QKeyGen}(sk)$ and submits R to the adversary.*
3. *$\mathcal{A}$ outputs a $(k+1)$-partite register R_{adv} and challenge messages m_ℓ^0, m_ℓ^1 for $\ell \in [k+1]$, 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. $b'_\ell \leftarrow \mathsf{U}_{quantum}(R_{\mathsf{adv}}[\ell], ct_\ell)$.*
   - *4.4. Check if $b'_\ell = b_\ell$.*
5. *The challenger outputs 1 if and only if all the checks pass.*

[11] Still polynomial since the adversary is QPT.

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* $\gamma = 0$*.*

Before moving onto the stronger definition, we need the following notation.

Definition 20 (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 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[12] measurements*, induced by* $\mathcal{D}$ *and* m_ℓ^0, m_ℓ^1, 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. *Run the universal circuit* $\mathsf{U}_{quantum}$ *on* (ρ, ct)*, let* b' *be the output.*
5. *Output 1 if* $b' = b$*. Otherwise, output* 0*.*

Observe that we can efficiently execute the above measurement[13] *for arbitrary given superpositions of* r *and* b *values. Therefore, by Section 3.7, 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_ℓ^0, m_ℓ^1, st *are omitted from the notation, they will be clear from the context. Unless otherwise specified, we will write* $\mathcal{D}$ *to denote the ciphertext distribution where we encrypt* m *as*

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

where pk *is part of* st*.*

Definition 21 (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* $\mathcal{A}$*.*

[12] While the measurement U described as-is is a general measurement, we will implicitly consider it as a measurement on input plus some ancilla in the state $|0\rangle^n$, since in its implementation we apply the unitary part of it on the input plus ancilla and finally make a measurement in the standard computational basis. In this case, the measurement is indeed projective, indexed by the output space of the challenge ciphertext distribution.

[13] 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* $R \leftarrow \mathsf{PKE.QKeyGen}(sk)$ *and submits* R *to the adversary.*
3. $\mathcal{A}$ *outputs a* $(k+1)$*-partite register* R_{adv} *and challenge messages* m_ℓ^0, m_ℓ^1 *for* $\ell \in [k+1]$*, 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 R_{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 25 ([CLLZ21])**.** *Suppose a public key encryption scheme with copy-protected keys satisfies CPA-style strong* γ*-anti-piracy (Definition 21). Then, it also satisfies CPA-style regular* γ*-anti-piracy (Definition 19).*

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 29.

Theorem 26 ([CLLZ21])**.** *Suppose a public key encryption scheme with copy-protected keys satisfies CPA-style regular* γ*-anti-piracy (Definition 19) 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.

6.2 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 10) 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 $\mathsf{CosetGen}$ 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,

While we assume exponential security of the above primitives for 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 in the underlying primitives.

Set $L(\lambda) = \lambda$ and therefore $c_L(\lambda) = 24 \cdot \lambda^3$ (see Theorem 19). 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 PMem_{K_1} is the following program.

   $\underline{\mathsf{PMem}_{K_1}(id, u_1, \ldots, u_{c_L(\lambda)}, r)}$

   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 $\left(\left|A_{i,s_i,s'_i}\right\rangle\right)_{i \in [c_L(\lambda)]}, ck, id$.

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

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.

$\underline{\mathsf{PCt}_{\mathsf{OPMem},cpk,K_2,r,m}(id, u_1, \ldots, u_{c_L(\lambda)})}$

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).

$\underline{\mathsf{PKE.Dec}(R_{\mathsf{key}}, ct)}$

1. Parse $((R_i)_{i\in[c_L(\lambda)]}, ck, id) = R_{\mathsf{key}}$ 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 $(R_i)_{i\in[c_L(\lambda)]}$.
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 27. 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 4. *Assuming subexponentially secure* $i\mathcal{O}$*, one-way functions and 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 2). F_1 and F_2 can be constructed based on one-way functions (Theorem 4). CCObf can be constructed based on $i\mathcal{O}$ and LWE (Theorem 8). □

6.3 Proof of Strong Anti-Piracy

In this section, we will prove Theorem 27. Throughout the proof, we will interpret identity strings for IBE, which are λ-bit strings, as integers in the set $\{0, 1, \ldots, 2^\lambda\}$. 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 by changing the way we sample the identity strings (in $\mathsf{PKE.QKeyGen}$) during each quantum key query 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 $\{1,\dots,2^\lambda-1\}$ *that has not appeared before*[14]. 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} 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 $\{0,1,\dots,2^\lambda-1\}$ k times gives k unique values from the set $\{1,\dots,2^\lambda-1\}$ 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 using threshold implementations $\mathsf{TI}_{\ell,\mathcal{D},1/2+\gamma}$, the challenger uses 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 14, 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. □

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 13 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 monogamy-of-entanglement game $\mathsf{MoE}-\mathsf{Coll}$ (Theorem 19). We will use projective and threshold implementations for various mixtures of measurements. 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 20. The particular distribution on the collection of projective measurements (induced by a challenge ciphertext distribution) will vary, and it will be denoted explicitly.

Notation. *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.*

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

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 above game $\mathcal{G}$ to $\mathsf{MoE}-\mathsf{Coll}$ with no loss of security, since the former is the same as the latter except that it includes an independent IBE instance that does not affect the game.

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

1. The challenger executes $pk, sk \leftarrow \mathsf{PKE.Setup}(1^\lambda)$ and submits pk to $\mathcal{A}'_0$.
2. For multiple 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 collusion-free way (as in Hyb_1), and submits it to $\mathcal{A}'_0$.
3. The adversary outputs a *bipartite* register R_{bip} and an index $j^* \in [k]$, where k is the number of queries it made.
4. For $\ell \in \{1, 2\}$, the challenger does the following.
   4.1. Sample $r_\ell \leftarrow \{0,1\}^{c_L(\lambda)}$.
   4.2. Run $\mathcal{A}'_\ell$ on $R_{\mathsf{bip}}[\ell]$, $(A_i^{j^*})_{i\in[c_L(\lambda)]}$ and r_ℓ to obtain a tuple of vectors $(v_{\ell,i})_{i\in[c_L(\lambda)]}$.
   4.3. 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 $j \in \{0, \ldots, k+1\}$ be efficient ciphertext distributions, which we will define later.

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

1. 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 R_{adv} be the $(k+1)$-partite register (with state σ) and $(m_\ell^0, m_\ell^1)_{\ell\in[k+1]}$ 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 $j^* \in \{1, \ldots, k\}$.
3. Apply $\mathsf{API}_{\ell,\mathcal{D}_0}^{\varepsilon,\delta}$ to all registers $R_{\mathsf{adv}}[\ell]$ for $\ell \in [k+1]$, let $b_{\ell,0}$ be the measurement outcomes.
4. Apply $\mathsf{API}_{\ell,\mathcal{D}_i}^{\varepsilon,\delta}$ in succession for $i = 1$ to j^* to $R_{\mathsf{adv}}[x]$, let $b_{x,i}$ be the measurement outcomes.
5. Apply $\mathsf{API}_{\ell,\mathcal{D}_i}^{\varepsilon,\delta}$ in succession for $i = 1$ to j^* to $R_{\mathsf{adv}}[y]$, let $b_{y,i}$ be the measurement outcomes.

6. Output

$$((R_{\mathsf{adv}}[x], j^*, x, y, (b_{\ell,0})_{\ell\in[k+1]}, (b_{x,i})_{i\in[j^*]}, (b_{y,i})_{i\in[j^*]}),$$
$$(R_{\mathsf{adv}}[y], j^*, x, y, (b_{\ell,0})_{\ell\in[k+1]}, (b_{x,i})_{i\in[j^*]}, (b_{y,i})_{i\in[j^*]},),$$
$$j^*).$$

For $j \in \{1, \ldots, k\}$, define $\mathcal{D}_j$ 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 $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}^{(j)}_{\mathsf{OPMem},cpk,K_2,r,m,id_j})$

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

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

   1. Run $\mathsf{OPMem}(id, u_1, \ldots, u_{c_L(\lambda)}, r)$. 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 $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}^{(k+1)}_{\mathsf{OPMem},cpk,K_2,r})$

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

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

   1. Run $\mathsf{OPMem}(id, u_1, \ldots, u_{c_L(\lambda)}, r)$. If it outputs 0, output $\perp$ and terminate.
   2. Output $\mathsf{IBE.Enc}(cpk, id, \top; F_2(K_2, id))$.
4. Output (OPCt, r).

Note that this distribution 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 index them as id_j.

Notation. *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. *Execute* $pk, sk \leftarrow \mathsf{PKE.Setup}(1^\lambda)$.
2. *Simulate the first two steps of* $\mathcal{A}'_0$ *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* $R_{\mathsf{adv}}, (m^0_\ell, m^1_\ell)_{\ell\in[k+1]}$ *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* $j^* \in \{1, \ldots, k\}$.
3. *Set* $pp = (x, y, j^*, (id_j)_{j\in[k+1]}, (m^0_\ell, m^1_\ell)_{\ell\in[k+1]}, pk)$.
4. *Sample* $b \leftarrow \{0,1\}$.
5. *Sample* $ct \leftarrow \mathcal{C}(pp, m^b_\ell)$.
6. *Output* $R_{\mathsf{adv}}, (b, ct), pp$.

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.

We will obtain efficient $\mathcal{A}'_1, \mathcal{A}'_2$ using Definition 5 such that $(\mathcal{A}'_0, \mathcal{A}'_1, \mathcal{A}'_2)$ wins $\mathcal{G}$ with probability

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

Finding a Simultaneous Jump

In the claim below, we will formalize the following fact. Observe the adversary only obtains k different identity keys for IBE. Therefore, informally, by 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, which is a contradiction by the monogamy-of-entanglement property.

Claim 5. *Let* τ *be the state of the bipartite register* $R_{\mathsf{adv}}[x, y]$ *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. 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 R_{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

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

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

Let ι be any quantum state of appropriate dimension. By Theorem 13, we have for all $\ell\in[k+1]$

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

Then, by Theorem 9, we have that if the outcome of $\mathsf{Pl}_{\ell,\mathcal{D}_{k+1}}$ 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,

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

Therefore, $\Pr\left[\left(\mathsf{APl}^{\varepsilon,\delta}_{\ell,\mathcal{D}_{k+1}}\right)\cdot\iota \geq \frac{1}{2}+\frac{2\gamma}{32}\right] \leq \exp(-\lambda)$. Now, if we apply $\mathsf{APl}^{\varepsilon,\delta}_{\ell,\mathcal{D}_{k+1}}$ 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

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

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 26). Then, by Theorem 12 and by above we get

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

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

In Theorem 12, 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 $\bigotimes_{\ell\in[k+1]}\mathsf{APl}^{\varepsilon,\delta}_{\ell,\mathcal{D}}\sigma$ and $\bigotimes_{\ell\in[k+1]}\mathsf{APl}^{\varepsilon,\delta}_{\ell,\mathcal{D}_1}\sigma$ when we define our collection of measurements as in Definition 20. 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

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

□

Now, we 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 $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}^{(j,\Delta,0)}_{\mathsf{OPMem},cpk,K_2,r,m,id_j+\Delta})$.

     $\underline{\mathsf{PCt}^{(j,\Delta,0)}_{\mathsf{OPMem},cpk,K_2,r,m,id_j+\Delta}(id, u_1, \ldots, u_{c_L(\lambda)})}$

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

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

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 10), 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 21 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 20 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* $j \in \{0,1,\ldots,k\}$, $\Delta \in \{0,1,\ldots,id_{j+1}-id_j-1\}$ *and* $\ell \in [k+1]$.

Proof. Essentially the same argument as in Claim 19 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* $j \in \{0,1,\ldots,k\}$
- $\mathsf{Exp}_{\mathcal{D}_j,\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(0,3)},\ell}$ *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 $i\mathcal{O}$.

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

Notation. *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 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 15, Theorem 14 and by definition of b_{x,j^*}, b_{y,j^*}, we have

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

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 12.

See the proof of Claim 5 for a remark on how to invoke Theorem 12. 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 12 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 12 and Theorem 16, 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 12 indeed applies. □

Extracting MoE Vectors

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 $R_{\mathsf{adv}}, (m^0_\ell, m^1_\ell)_{\ell\in[k+1]}$ 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 $R_{\mathsf{adv}}[\ell]$ 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 $R_{\mathsf{adv}}[x]$, 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 $R_{\mathsf{adv}}[y]$, let $b_{y,i}$ be the measurement outcomes.

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

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

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

6. Output $R_{\mathsf{adv}}[x], (b, ct), pp$.

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 former using the auxiliary state of the latter.

By Claim 14, we have

1. $\Pr\left[\mathsf{PI}_{x,\mathcal{D}'} \cdot \tau[1] \le b_{x,j^*} - \frac{3\gamma}{32k}\right] \le 2^{-2\lambda} \cdot 2^{-4\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$, 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 13 and Theorem 12, 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 5 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 $R_{\mathsf{adv}}, (m^0_\ell, m^1_\ell)_{\ell\in[k+1]}$ 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 $R_{\mathsf{adv}}[\ell]$ 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 $R_{\mathsf{adv}}[x]$, 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 $R_{\mathsf{adv}}[y]$, let $b_{y,i}$ be the measurement outcomes.
3. Set $pp = (x, y, j^*, (id_j)_{j\in[k+1]}, (m^0_\ell, m^1_\ell)_{\ell\in[k+1]}, pk)$.
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 $g(v_1, \ldots, v_{c_L(\lambda)}) = (g_1(v_1)||\ldots||g_{c_L(\lambda)}(v_{c_L(\lambda)}))$.
   5.8. Set $y = y_1||\ldots||y_{c_L(\lambda)}$.
6. Output (g, y, ct^*) as the compute-and-compare program and
$$(R_{\mathsf{adv}}[x], pp, r, m^b_x, id_{j^*}, b)$$
as the auxiliary information.

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

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

1. Parse $(R, pp, r, m^b_x, id_{j^*}, b) = R_{\mathsf{aux}}$.
2. Parse $(x, y, j^*, (id_j)_{j\in[k+1]}, (m^0_\ell, m^1_\ell)_{\ell\in[k+1]}, pk) = pp$.
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 $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt})$.

$\underline{\mathsf{PCt}(id, u_1, \ldots, u_{c_L(\lambda)})}$

Hardcoded: $\mathsf{OPMem}, cpk, K_2\{id_{j^*}\}, r, m^b_x, id_{j^*}, P$

(a) If $id = id_{j^*}$, output the output of $P(u_1, \ldots, u_{c_L(\lambda)})$ 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 $id < id_{j^*} + 1$, set $a = \top$. Otherwise, set $a = m^b_x$.

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

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

It is easy to see that $\mathcal{A}_{CC}(\mathsf{CCObf.Obf}(g, y, ct^*), R_{\mathsf{aux}})$ corresponds to $\mathcal{A}_{dist}(\mathsf{Exp}_{\mathcal{D}',x})$ while $\mathcal{A}_{CC}(\mathsf{CCObf.Sim}(1^\lambda, |g$ 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 8 there exists an adversary $\mathcal{M}_1$ that can extract vectors u_i such that $g((u_i)) = y$, using the quantum auxiliary information defined above and the description of g. Note that g can be computed efficiently given r and $(A^*_i, s^*_i, s'^*_i)_{i\in[c_L(\lambda)]}$, which are indeed provided to $\mathcal{A}'_1$ in $\mathcal{G}$. Similarly, $\mathcal{A}'_1$ can compute R_{aux} from its input provided by $\mathcal{A}'_0$. Therefore, we set $\mathcal{A}'_1 = \mathcal{M}_1$ and 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 $(R_{\mathsf{adv}}[x], pp, r, m^b_x, id_{j^*}, b)$ 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 $R_{\mathsf{adv}}[y], (b, ct), pp$.

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

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

This first claim follows from Claim 14, Theorem 6, and the fact that extraction on the first register succeeds with probability $\ge 2^{-\lambda^{0.2\cdot C_{\mathsf{MoE.Coll}}}}$. 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 $\frac{\Pr[E|G]\cdot\Pr[G]}{\Pr[E]} \geq \Pr[E|G]\cdot\Pr[G] \geq \Pr[E|G]\cdot\frac{1}{\mathrm{poly}(\lambda)}$. 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 6, since $\Pr[G] > 1/\mathrm{poly}(\lambda)$. 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 $2^{3\lambda}\cdot 2^{2\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$-time adversaries where $\nu = 2^{-2\lambda-1}\cdot 2^{-2\lambda^{0.3C_{\mathsf{MoE.Coll}}}}$. Then, by Theorem 12, 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 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 $\mathsf{MoE}-\mathsf{Coll}$.

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

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

Proof. $\mathcal{A}''_0$ simulates both the challenger of $\mathcal{G}$ and the adversary $\mathcal{A}'_0$ as follows. It first samples $cpk, csmk \leftarrow \mathsf{IBE.Setup}(1^\lambda)$. Then, it sets $pk = (cpk, \mathsf{OPMem})$ where it obtains OPMem from its challenger. Then, whenever $\mathcal{A}'_0$ makes a key query, it samples an identity string id in a collusion-free way as in $\mathcal{G}$ and queries its own challenger for the coset state associated with id. It also samples ck as in $\mathsf{PKE.QKeyGen}$ using $cmsk$, and submits the coset state, id and ck to $\mathcal{A}'_0$. Finally, when $\mathcal{A}'_0$ yields a bipartite register and an index j^*, $\mathcal{A}''_0$ outputs both.

We define $\mathcal{A}''_1$ to simulate $\mathcal{A}'_1$ 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 $\mathsf{MoE}-\mathsf{Coll}$ 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 19.

7 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 in this 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. Further, the freeloader adversaries are not allowed to query for more keys after getting their challenge ciphertexts.

7.1 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 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 them each with a challenge ciphertext. Finally, after receiving the challenge ciphertexts, freeloaders can query for more keys, and they output their guess at the end.

For the challenge message pair m^0, m^1, we will require that the pirate can obtain a *classical* key for a function f only if it satisfies $f(m^0) = f(m^1)$. Similarly, the freeloader can obtain a key for f only under this condition. However, we will not require anything for functional keys that were only obtained in 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. Therefore, we do not need to separately require that a copy-protected key for f allows no more than evaluation of the encrypted message under f, which is already implied by the regular functional encryption security.

Definition 22 (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 3) with the following additional algorithm and guarantee.*

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

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

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

As discussed in Section 6, 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 23 (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* 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 $R_f \leftarrow \mathsf{FE.QKeyGen}(sk_f)$ and submits R_f to the adversary.*

3. *The adversary outputs a pair of challenge messages m^0, m^1 and a $(k+1)$-partite register R_{adv} (where k is the number of queries of the type* 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.*

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* CLASSICAL *or* PROTECTED*. For each f, the challenger does the following. It first computes $sk_f \leftarrow \mathsf{FE.KeyGen}(msk, f)$. Then, it adds f to $\mathcal{F}_\ell$.*

   If the query type is CLASSICAL*, the challenger submits sk_f to the adversary ℓ. If the query type is* PROTECTED*, the challenger computes $R_f \leftarrow \mathsf{FE.QKeyGen}(sk_f)$ 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 $b'_\ell = b_\ell$ 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 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 CLASSICAL.

Finally, note that as opposed to our public-key encryption definition, the piracy adversary is not allowed to choose different challenge message pairs for each freeloader. Looking ahead, this is partially an artifact of our proof technique. In our proof, we will need to remove Query Phase 2 from the game, since our techniques are restricted to non-interactive freeloaders. To achieve this without weakening our security model, we will instead define a scheme where the freeloader adversaries receive a punctured key $pmsk$ that allows them to obtain secret keys for functions f that satisfy $f(m^0) = f(m^1)$. If we allowed different challenge messages for each freeloader, then their punctured keys would also be different. While they are not allowed to communicate, this situation might possibly allow them to use these punctured keys to decrypt their ciphertexts, since entanglement provides advantage over classical strategies in non-local games (e.g CHSH game [CHSH69]). We leave it as an open question to construct schemes where the freeloaders are allowed to have different challenge message pairs.

7.2 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 14) and deterministic KeyGen that satisfies strong punctured key correctness (Definition 13), 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 $\mathsf{CosetGen}$ 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,

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.

   $\underline{\mathsf{PCt}_{\mathsf{OPMem},cpk,K_2,r,m}(id||f, u_1, \ldots, u_{c_L(\lambda)})}$

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

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

5. Output (OPCt, r).

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

1. Parse $((R_i)_{i\in[c_L(\lambda)]}, ck, id, f) = R_{\mathsf{key}}$ 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 $(R_i)_{i\in[c_L(\lambda)]}$.
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 28. FE *satisfies* γ*-anti-piracy (Definition 23) for any inverse polynomial* γ*.*

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

Corollary 5. *Assuming subexponentially secure* $i\mathcal{O}$*, one-way functions and LWE, there exists a public-key functional encryption scheme that satisfies anti-piracy security against unbounded collusion.*

7.3 Proof of Anti-Piracy

Proof will closely follow the strong anti-piracy security proof for our public-key encryption construction in Section 6.3, which crucially relies on projective implementations to simultaneously extract vectors from all registers to reduce 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 5.2.

Definition 24 (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* $R_f \leftarrow \mathsf{FE.QKeyGen}(sk_f)$ *and submits* R_f *to the adversary.*
3. *The adversary outputs a pair of challenge messages* m^0, m^1 *and a* $(k+1)$*-partite register* R_{adv} *(where* k *is the number of queries of the type* $\mathsf{PROTECTED}$*), each part of the register being a freeloader adversary that will be executed using a universal circuit.*
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 submits* $ct_\ell, pmsk$ *to the* ℓ*-th freeloader to obtain a guess* b'_ℓ. *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,*

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

Definition 25 (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 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, 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. *Run the universal circuit* $\mathsf{U}_{quantum}$ *on* $(\rho, pmsk, ct)$, *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 3.7, 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, st, *omitted in the notation, will be clear from the context. Unless otherwise specified, we will write* $\mathcal{D}$ *to denote the distribution where the ciphertext is sampled as*

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

and $pmsk$ *is sampled as in Definition 24, where* pk *is part of* st.

Definition 26 (CPA-Style Strong Anti-Piracy Security for FE)**.** *Consider the following game between the challenger and an adversary* $\mathcal{A}$.

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

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 $R_f \leftarrow \mathsf{FE.QKeyGen}(sk_f)$ *and submits* R_f *to the adversary.*

3. *The adversary outputs a pair of challenge messages* m^0, m^1 *and a* $(k+1)$*-partite register* R_{adv} *(where* k *is the number of queries of the type* PROTECTED*), each part of the register being a freeloader adversary that will be executed using a universal circuit.*

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,*

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

We first prove that the stronger definition implies the regular anti-piracy security (Definition 23).

Theorem 29. *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.

As discussed before, there exists a projective implementation for each decryption test the challenger performs above for each register, where instead of applying the universal circuit and comparing the output to b'_ℓ, if we apply the projective implementation 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[15]. 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

$$\Pr\left[\forall \ell \in [k+1] \;\; p_\ell \geq \frac{1}{2} + \gamma(\lambda)\right] \leq \mathsf{negl}(\lambda).$$

[15]Note that the joint distribution is still 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.

Then, by above,

$$\begin{aligned}\Pr[\mathsf{FEAntiPiracyNI}(\lambda,\mathcal{A})=1] =& \mathbb{E}[p_1\cdots p_\ell]\\ =& \Pr\left[\forall \ell\in[k+1]\ \ p_\ell\geq\frac{1}{2}+\gamma(\lambda)\right]\cdot\mathbb{E}[p_1\cdots p_\ell|\forall\ell\in[k+1]\ \ p_\ell\geq\frac{1}{2}+\gamma(\lambda)]+\\ & \Pr\left[\exists \ell\in[k+1]\ \ p_\ell<\frac{1}{2}+\gamma(\lambda)\right]\cdot\mathbb{E}[p_1\cdots p_\ell|\exists\ell\in[k+1]\ \ p_\ell<\frac{1}{2}+\gamma(\lambda)]\\ \leq& \mathsf{negl}(\lambda)\cdot 1+1\cdot(\frac{1}{2}+\gamma(\lambda)).\end{aligned}$$

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. □

Reducing to Monogamy-of-Entanglement

Lemma 11. FE *satisfies strong γ-anti-piracy security for any inverse polynomial γ.*

Proof of this lemma is almost identical to the proof of security of our public-key encryption scheme. Therefore, we will omit the proofs of some claims.

Throughout the proof, we will interpret identity strings for IBE, which are $Q(\lambda)+\lambda$-bit strings, as integers in the set $\{0,1,\ldots,2^{Q(\lambda)+\lambda}\}$.

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 $\mathsf{FEStrongAntiPiracy}(\lambda,\gamma(\lambda),\mathcal{A})$.

Define Hyb_1 by modifying Hyb_0 as follows. When generating functional keys, we sample the identity strings in a collusion-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 6.3. □

Notation. *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 $R_f \leftarrow \mathsf{FE.QKeyGen}(sk_f)$ 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 R_{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 $R_{\mathsf{bip}}[\ell]$, $(A_i^{j^*})_{i\in[c_L(\lambda)]}$ 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 $v_{\ell,i} \in (A^{j^*})_i^{\perp} + s_i^{\prime j^*}$ 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[16], we will assume that all 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 R_{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 $1 \leq x < y \leq k+1$ and $j^* \in \{1, \ldots, k\}$.
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 $R_{\mathsf{adv}}[\ell]$ 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 $R_{\mathsf{adv}}[x]$, let $b_{x,i}$ be the measurement outcomes.

[16] 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 $f(m_0) \neq f(m_1)$.

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

7. Output

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

For $j \in \{1, \ldots, k\}$, 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 $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}^{(j)}_{\mathsf{OPMem},cpk,K_2,r,m,id_j||f_j})$

   $\mathsf{PCt}^{(j)}_{\mathsf{OPMem},cpk,K_2,r,m^b,m_{1-b},id_j||f_j}(id||f, u_1, \ldots, u_{c_L(\lambda)})$

   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 $\bot$ 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 $\bot$ 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 24. 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 $(\mathsf{OPCt},r),pmsk$.

Notation. *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 R_{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 $1\le x<y\le k+1$ and $j^*\in\{1,\ldots,k\}$.*
   - 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 $R_{\mathsf{adv}},(b,ct,pmsk),pp$.*

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 $R_{\mathsf{adv}}[x,y]$ 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 $\le 1/2$. However, this is indeed true, since $\mathcal{D}_{k+1}$ is encoding m^{1-b} while the challenge bit is b. □

Notation. *When we refer to $(id||f+\Delta)$ as a function, we will mean the function f' which the function whose description is the last $Q(\lambda)$ bits of $(id||f+\Delta)$.*

Now, we define some intermediary distributions. Define the following for all $j \in \{0,1,\ldots,k\}$ and $\Delta \in \{0,1,\ldots,id_{j+1}||f_{j+1} - id_j||f_j - 1\}$. 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 $j \in \{0,1,\ldots,k\}$. Also note that $\mathcal{D}_j^{(0,0)}$ is exactly the same as $\mathcal{D}_j$ for $j \in [k]$.

- $\underline{\mathcal{D}_j^{(\Delta,0)}}$:
  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 $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}^{(j,\Delta,0)}_{\mathsf{OPMem},cpk,K_2,r,m,id_j+\Delta})$.

     $\mathsf{PCt}^{(j,\Delta,0)}_{\mathsf{OPMem},cpk,K_2,r,m^b,m^{1-b},id_j||f_j+\Delta}(id||f,u_1,\ldots,u_{c_L(\lambda)})$

     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 $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}^{(j,\Delta,1)}_{\mathsf{OPMem},cpk,K_2\{id_j+\Delta\},r,m,id_j+\Delta,ct^*})$.

     $\mathsf{PCt}^{(j,\Delta,1)}_{\mathsf{OPMem},cpk,K_2\{id_j+\Delta\},r,m,id_j+\Delta,ct^*}(id||f,u_1,\ldots,u_{c_L(\lambda)})$

     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 $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}^{(j,\Delta,2)}_{\mathsf{OPMem},cpk,K_2\{id_j+\Delta\},r,m,id_j+\Delta,ct^*})$.

> $\mathsf{PCt}^{(j,\Delta,2)}_{\mathsf{OPMem},cpk,K_2\{id_j||f_j+\Delta\},r,m^b,m^{1-b},id_j||f_j+\Delta,ct^*}(id||f,u_1,\ldots,u_{c_L(\lambda)})$
>
> **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 $(\mathsf{OPCt}, r), pmsk$.

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

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 $(A_i^*, s_i^*, s_i'^*) = F_1(K_1, id_j||f_j+\Delta)$.
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 $y_i = g_i(s_i'^*)$ 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 $\mathsf{OPCt} \leftarrow i\mathcal{O}(\mathsf{PCt}^{(j,\Delta,3)}_{\mathsf{OPMem},cpk,K_2\{id_j\},r,m,id_j+\Delta,\mathsf{OCC}})$.

> $\mathsf{PCt}^{(j,\Delta,3)}_{\mathsf{OPMem},cpk,K_2\{id_j+\Delta\},r,m,id_j||f_j+\Delta,\mathsf{OCC}}(id||f,u_1,\ldots,u_{c_L(\lambda)})$
>
> **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

$$ct^* = \mathsf{IBE.Enc}(cpk, id_j||f_j + \Delta, (id_j||f_j + \Delta)(m^{1-b}); F_2(K_2, id_j||f_j + \Delta)).$$

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

Claim 19. $\mathsf{Exp}_{\mathcal{D}_j^{(\Delta,0)},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(\Delta,1)},\ell}$ *for all* $j \in \{0, 1, \ldots, k\}$, $\Delta \in \{0, 1, \ldots, id_{j+1} - id_j - 1\}$ *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 20. $\mathsf{Exp}_{\mathcal{D}_j^{(\Delta,1)},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(\Delta,2)},\ell}$ *for all* $j \in \{0, 1, \ldots, k\}$, $\Delta \in \{0, 1, \ldots, id_{j+1} - id_j - 1\}$ *and* $\ell \in [k+1]$.

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

Claim 21. $\mathsf{Exp}_{\mathcal{D}_j^{(\Delta,2)},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(\Delta,3)},\ell}$ *for all* $j \in \{0, 1, \ldots, k\}$, $\Delta \in \{0, 1, \ldots, id_{j+1} - id_j - 1\}$ *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 22. $\mathsf{Exp}_{\mathcal{D}_j^{(\Delta,3)},\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(\Delta,4)},\ell}$ *if*

- $j \in \{1, \ldots, k\}$ *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. 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 $pmsk \leftarrow i\mathcal{O}(\mathsf{PKey}_{cmsk', K_1\{id_{j^*}||f_{j^*}\}})$.

$\underline{\mathsf{PKey}_{cmsk', K_1\{id_{j^*}||f_{j^*}\}}(id||f)}$

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 $id_1||f_1, id_2||f_2, \ldots, id_k||f_k$, 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* $j \in \{0, 1, \ldots, k\}$, $\Delta \in \{0, 1, \ldots, id_{j+1} - id_j - 1\}$ *and* $\ell \in [k+1]$.

Proof. Essentially the same argument as in Claim 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* $j \in \{0,1,\ldots,k\}$, $\Delta \in \{0,1,\ldots,id_{j+1}-id_j-1\}$ *and* $\ell \in [k+1]$.

Proof. Essentially the same argument as in Claim 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* $j \in \{0,1,\ldots,k\}$, $\Delta \in \{0,1,\ldots,id_{j+1}-id_j-1\}$ *and* $\ell \in [k+1]$.

Proof. Essentially the same argument as in Claim 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* $j \in \{0,1,\ldots,k\}$
- $\mathsf{Exp}_{\mathcal{D}_j,\ell} \approx^c_{\nu(\lambda)} \mathsf{Exp}_{\mathcal{D}_j^{(0,3)},\ell}$ *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. □

Notation. *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,*

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

Proof. 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*

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

Proof. It is straightforward to reduce $\mathcal{G}$ to the stronger version of $\mathsf{MoE}-\mathsf{Coll}$ shown secure in the proof of Theorem 19. See also Claim 16. □

The above constitutes a contradiction by Theorem 19, therefore this completes the security proof.

8 Acknowledgements

We thank Jiahui Liu for helpful discussions.

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.

[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.

[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.

[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.

[BV18] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *J. ACM*, 65(6), nov 2018.

[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.

[ÇGLZR23] Alper Çakan, Vipul Goyal, Chen-Da Liu-Zhang, and João Ribeiro. Unbounded leakage-resilience and leakage-detection in a quantum world. Cryptology ePrint Archive, Paper 2023/410, 2023. https://eprint.iacr.org/2023/410.

[CHSH69] John F Clauser, Michael A Horne, Abner Shimony, and Richard A Holt. Proposed experiment to test local hidden-variable theories. *Physical review letters*, 23(15):880, 1969.

[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.

[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.

[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.

[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.

[Zha19] Mark Zhandry. Quantum lightning never strikes the same state twice. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 408–438, Cham, 2019. Springer International Publishing.

[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 5

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 Theorem 6

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 $\Pi_1 = \sum_{j\in\mathcal{J}'}|v_j\rangle$ and $\Pi_1' = \sum_{k\in\mathcal{K}'}|w_k\rangle$ for some subsets $\mathcal{J}' \subseteq \mathcal{J}, \mathcal{K}' \subseteq \mathcal{K}$. Since $\mathrm{Tr}\{\Pi_1 \otimes \Pi_1'\rho\}$, 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 $|\phi\rangle = \sum_{j\in\mathcal{J}',k\in\mathcal{K}'} \alpha_{j,k}(M_i|v_j\rangle)\otimes|w_k\rangle$. When we apply $I\otimes\Pi_1'$ to $|\phi\rangle$, we get $|\phi\rangle$ again. Hence, $\mathrm{Tr}\left\{\Pi_1' \frac{|\phi\rangle\langle\phi|}{\||\phi\rangle\|^2}\right\} = 1$, 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 obtaining the outcome $(1,1)$. Note that ρ' satisfies the claim with $\varepsilon = 0$. Then, by Lemma 2, $\|\rho - \rho'\|_{Tr} \leq \sqrt{\varepsilon}$. 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 $\sqrt{\varepsilon}/p_i$. Hence, invoking the sub-claim for ρ' with $\varepsilon = 0$ and using the trace distance bound, we get the result.

A.3 Proof of Theorem 7

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.4 Proof of Theorem 12

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.4.1 Technical Lemmata

Lemma 12. *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 13. *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 $i \in \{0, 1, \ldots, s(\lambda)\}$, 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 $j \le i-1$ 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 $\ge \nu(\lambda)$, which is a contradiction.

Finally, triangle inequality yields the claim. □

Lemma 14 ([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*

$$\left|\Pr\left[B^{\Pi}() = 1\right] - \Pr\left[B^{G\circ F}() = 1\right]\right| \le O(Q^3/s + Q^3/|A|).$$

Lemma 15 ([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.4.2 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 $\left(\bigotimes_{\ell\in[i]} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{B}_\ell''^0(pp))}\right) \otimes \left(\bigotimes_{\ell\in\{i+1,\ldots,k\}} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{B}_\ell'^0(pp))}\right)$ 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 $\left(\bigotimes_{\ell\in[i]} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{B}_\ell'''^0(pp))}\right) \otimes \left(\bigotimes_{\ell\in\{i+1,\ldots,k\}} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{B}_\ell''^0(pp))}\right)$ where we define $\mathcal{B}_\ell'''^0(pp;r) = \mathcal{B}_\ell^0(pp;G(E(r)))$.

$\mathsf{Hyb}_{5,i}$ **for** $i \in \{0,1,\ldots,k-1\}$**:** We now apply $\left(\bigotimes_{\ell\in[i]} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{B}_\ell'''^1(pp))}\right) \otimes \left(\bigotimes_{\ell\in\{i+1,\ldots,k\}} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{B}_\ell'''^0(pp))}\right)$.

$\mathsf{Hyb}_{6,k-i+1}$ **for** $i \in [k]$**:** We now apply $\left(\bigotimes_{\ell\in[i]} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{B}_\ell'''^1(pp))}\right) \otimes \left(\bigotimes_{\ell\in\{i+1,\ldots,k\}} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{B}_\ell''^1(pp))}\right)$ 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 $\left(\bigotimes_{\ell\in[i]} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{B}_\ell''^1(pp))}\right) \otimes \left(\bigotimes_{\ell\in\{i+1,\ldots,k\}} \mathsf{API}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{B}_\ell'^1(pp))}\right)$ 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 16. $|\mathsf{Hyb}_0 - \mathsf{Hyb}_1| \leq \nu(\lambda)$

Lemma 17. $\mathsf{Hyb}_1 \equiv \mathsf{Hyb}_2$ *and* $\mathsf{Hyb}_8 \equiv \mathsf{Hyb}_9$.

Proof. Observe that $\mathcal{B}'^b_\ell(pp)$ and $\mathcal{B}^b_\ell(pp)$ are exactly the same distribution. The result follows from Lemma 12. □

Lemma 18.

- $|\mathsf{Hyb}_2 - \mathsf{Hyb}_{3,1}| \leq O(Q^3/s)$.
- $|\mathsf{Hyb}_{3,i} - \mathsf{Hyb}_{3,i+1}| \leq O(Q^3/s)$ *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 14. □

Lemma 19.

- $\mathsf{Hyb}_{3,k} \equiv \mathsf{Hyb}_{4,1}$.
- $\mathsf{Hyb}_{4,i} \equiv \mathsf{Hyb}_{4,i+1}$ *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 15. □

Lemma 20.

- $|\mathsf{Hyb}_{5,k-1} - \mathsf{Hyb}_{6,0}| \leq \nu(\lambda) \cdot s(\lambda)$.
- $|\mathsf{Hyb}_{5,i-1} - \mathsf{Hyb}_{5,i}| \leq \nu(\lambda) \cdot s(\lambda)$ *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 $\mathrm{poly}(\lambda) \cdot k \cdot s$. The result then follows from Lemma 13. □

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 $Q = \mathrm{poly}(\lambda)$.

A.5 Proof of Theorem 15

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.

We now prove the fourth point. Let p_i denote

$$\mathrm{Tr}\left[\left(\bigotimes_{\ell\in[i]} \mathsf{TI}_{\eta_\ell-\varepsilon}(\mathcal{P}_{\ell\mathcal{D}_\ell})\right) \otimes \left(\bigotimes_{\ell\in\{i+1,\ldots,k\}} \mathsf{ATI}^{\varepsilon,\delta}_{\mathcal{P}_\ell,\mathcal{D}_\ell,\eta_\ell}\right)\rho\right]$$

Fourth point follows from a simple hybrid lemma.

A.6 Proof of Theorem 15

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 $\Pr\left[\forall \ell \in [k] \;\; (\vec{p'})_\ell \leq (\vec{p})_\ell + 2\varepsilon\right] \geq 1 - 2\cdot k\cdot\delta$ by Theorem 7.

A.7 Proof of Theorem 16

Follows from a simpe hybrid lemma.