arXiv:2507.19032v1 [cs.CR] 25 Jul 2025

How to Copy-Protect Malleable-Puncturable Cryptographic Functionalities Under Arbitrary Challenge Distributions

Alper Çakan*
Carnegie Mellon University
alpercakan98@gmail.com

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

Abstract

A quantum copy-protection scheme (Aaronson, CCC'09) encodes a functionality into a quantum state such that given this state, no efficient adversary can create two (possibly entangled) quantum states that are both capable of running the functionality. There has been a recent line of works on constructing provably-secure copy-protection schemes for general classes of schemes in the plain model, and most recently the recent work of Çakan and Goyal (IACR Eprint, 2025) showed how to copy-protect all cryptographically puncturable schemes with pseudorandom puncturing points.

In this work, we show how to copy-protect even a larger class of schemes. We define a class of cryptographic schemes called *malleable-puncturable* schemes where the only requirement is that one can create a circuit that is capable of answering inputs at points that are *unrelated* to the challenge in the security game but does not help the adversary answer inputs *related* to the challenge. This is a flexible generalization of puncturable schemes, and can capture a wide range of primitives that was not known how to copy-protect prior to our work.

Going further, we show that our scheme is secure against arbitrary high min-entropy challenge distributions whereas previous work has only considered schemes that are punctured at pseudorandom points.

*Part of this work was done while the author was an intern at NTT Research.

Contents

1 Introduction

Starting with the seminal work of Wiesner [Wie83], a long line of research has shown that quantum information can be extremely useful in cryptography. One of the main areas where quantum information helps cryptography is achieving a task that is classically impossible task no matter what cryptographic assumptions are made. This line of applications of quantum information is also called *quantum protection*. This notion can further be divided into four main categories: *copy-protection* ([Aar09]), *secure leasing* ([AL21]), *unbounded/LOCC leakage-resilience* ([ÇGLZR24]) and *intrusion-detection* ([ÇGLZR24]).

A recent line of works ([CG24b, AB24]) have focused on constructing quantum copy-protection schemes for general classes of functionalities, culminating in the recent work of Çakan and Goyal ([ÇG25]) who showed how to copy-protect in the plain model all cryptographically puncturable schemes with pseudorandom puncturing points. In this work, we further advance the class of functionalities that can be copy-protected: We define a class of cryptographic schemes called *malleable-puncturable* schemes where the only requirement is that one can create a circuit that is capable of answering inputs at points that are *unrelated* to the challenge in the security game but does not help the adversary answer inputs *related* to the challenge. This is an extremely flexible generalization of puncturable schemes, and can capture a wide range of primitives that was not known how to copy-protect prior to our work.

Theorem 1 (Corollary 1, informal)**.** *For any functionality* F *that satisfies* malleable-puncturing *security (Definition 4), there exists a quantum copy-protection scheme for* F *secure against arbitrary high min-entropy challenge distributions; assuming subexponentially secure iO and one-way functions.*

2 Preliminaries

2.1 Notation

When S is a set, we write $x \leftarrow S$ to mean that x is sampled uniformly at random from S. When $\mathcal{D}$ is a distribution, we write $x \leftarrow \mathcal{D}$ to mean that x is sampled from $\mathcal{D}$. Finally, we write $x \leftarrow \mathcal{B}(\mathsf{R})$ or $\mathcal{B}(a)$ to mean that x is set to a sample as output by the quantum or classical randomized algorithm $\mathcal{B}$ run on R or a. We use similar notation for quantum registers, e.g., $\mathsf{R} \leftarrow \mathcal{B}(a)$.

We write *QPT* to mean a stateful (unless otherwise specified) *quantum polynomial time*. We write R to mean a quantum register, which keeps a quantum state that will evolve when the register is acted on, and it can be entangled with other registers. We write $\mathsf{R} \leftarrow \rho$ to mean that the register R is initialized with a sample from the quantum distribution (i.e. mixed state) ρ. We refer the reader to [NC10] for a preliminary on quantum information and computation.

Unless otherwise specified, all of our cryptographical assumptions are implicitly post-quantum, e.g., *one-way functions* means *post-quantum secure one-way functions*.

2.2 Cryptography

2.2.1 Puncturable Pseudorandom Functions

In this section, we recall puncturable pseudorandom functions.

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

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

We require the following.

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

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

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 $K \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ and $K_S \leftarrow \mathsf{PRF.Puncture}(K, S)$*
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 2 ([SW14, Zha12])**.** *Let $n(\cdot), m(\cdot), e(\lambda), k(\lambda)$ be efficiently computable functions.*

- *If (post-quantum) one-way functions exist, then there exists a (post-quantum) puncturable PRF with input space $\{0,1\}^{m(\lambda)}$ and output space $\{0,1\}^{n(\lambda)}$.*
- *If we assume subexponentially-secure (post-quantum) one-way functions exist, then for any $c > 0$, there exists a (post-quantum) $2^{-\lambda^c}$-secure[1] puncturable PRF against subexponential time adversaries with input space $\{0,1\}^{m(\lambda)}$ and output space $\{0,1\}^{n(\lambda)}$.*

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

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

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

2.3 Quantum Information

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 $1 - \epsilon$, we have*

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

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

Theorem 3 ([ÇG24a])**.** *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*

$$\mathrm{Tr}\left\{\Pi_1 \otimes \Pi'_1 \rho\right\} \geq 1 - \epsilon.$$

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

$$\mathrm{Tr}\left\{\Pi'_1 \tau\right\} \geq 1 - \frac{3\sqrt{\epsilon}}{2p_i}.$$

Theorem 4 (Implementation Independence of Measurements on Bipartite States ([ÇG24a]))**.** *Let $\Lambda = \{M_i\}_{i \in \mathcal{I}}, \Lambda' = \{E_i\}_{i \in \mathcal{I}}$ be two general measurements whose POVMs are equivalent, that is, $M_i^\dagger M_i = E_i^\dagger E_i$ for all $i \in \mathcal{I}$.*

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

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

3 Definitional Work

3.1 Malleable-Puncturable Schemes

We first recall the following template from [ÇG25] that captures most cryptographic schemes along with their security games.

Definition 3 (Cryptographic Scheme with Security Game ([ÇG25]))**.** *A cryptographic scheme with security game is a tuple of efficient algorithms* $(\mathsf{Eval}, \mathsf{Ver}, \mathsf{Chal}, \mathsf{SampCh})$*, defined as follows.*

- Eval *is a (possibly quantum) algorithm that receives as input some key* $k \in \mathcal{K}$ *and some input* $z \in \mathcal{Z}$*, and outputs some* $y \in \mathcal{Y}$*,*
- Chal *is an (possibly quantum) interactive algorithm that represents the setup of the security game such that at the end of the interaction with an adversary, it outputs a key* $k \in \mathcal{K}$ *and challenger state*[2] *st,*
- SampCh *is a classical input-output (possibly quantum) sampler that receives as input st, and outputs a challenge ch (which will be sent to the adversary) and the answer key*[3] *ak,*
- Ver *is a (possibly quantum) verifier that receives as input the answer key ak and an alleged answer* ans'*, and it outputs* TRUE *or* FALSE*.*

We define meaningfulness and security as below.

$p^{\mathcal{F}}_{triv}$-Security: *For any QPT adversary* $\mathcal{A}$*, we have* $\Pr\left[\mathsf{SecurityGame}_{\mathcal{A}}(1^\lambda) = 1\right] \leq p^{\mathcal{F}}_{triv}(\lambda) + \mathsf{negl}(\lambda)$ *where the security game* $\mathsf{SecurityGame}$ *is defined as follows*[4]*.*

$\mathsf{SecurityGame}_{\mathcal{A}}(1^\lambda)$

1. $\mathsf{Chal}(1^\lambda)$ *and* $\mathcal{A}(1^\lambda)$ *interact*[5]*,* Chal *outputs k and st.*
2. *The challenger runs* $ak, ch \leftarrow \mathsf{SampCh}(st)$*.*
3. *The adversary* $\mathcal{A}$ *receives ch and outputs* ans'*.*
4. *The challenger outputs* 1 *if* $\mathsf{Ver}(ak, ans') = \mathsf{TRUE}$*, otherwise outputs* 0*.*

Meaningfulness: *There exists an efficient (possibly quantum) algorithm* $\mathcal{B}$ *such that* $\mathsf{Ver}(ak, \mathcal{B}(k, ch)) = \mathsf{TRUE}$ *with probability* $1 - \mathsf{negl}(\lambda)$*. where the values are sampled as in the experiment above.*

We now define the notion of malleable-puncturable schemes.

Definition 4 (Malleable-Puncturable Scheme)**.** *A malleable-puncturable scheme is a cryptographic scheme* $\mathcal{CS} = (\mathsf{Eval}, \mathsf{Ver}, \mathsf{Chal}, \mathsf{SampCh})$ *with the following additions and modifications.*

- *The key k is parsed as* $(C, aux, Q_{\mathsf{rel}})$ *where C is a classical circuit.*
- Eval *algorithm* $\mathsf{Eval}(k, z)$ *is equivalent to* $\mathsf{Eval}^C(aux, z)$ *(C can be called possibly in superposition).*

[2] This represents the information needed to sample a challenge. For example, this could be the challenge messages m_0, m_1 in a CPA security game for encryption.

[3] This key here represents the information that is needed to verify adversary's answer. For example, in a CPA security game, this will be the challenge bit b where the challenge ciphertext ct is an encryption of m_b.

[4] Here, $p^{\mathcal{F}}_{triv}$ denotes the baseline/trivial success probability. For example, for a chosen plaintext security game, we will have $p^{\mathcal{F}}_{triv} = 1/2$.

[5] During this interaction, $\mathcal{A}$ may learn some information related to the secret s. For example, in a public-key encryption security game, $\mathcal{A}$ will learn the public-key

- SampCh *consists of two steps:* $\mathcal{D}_{\mathsf{inp}}(st)$ *which outputs* x*, and* $\mathsf{SampChFromInp}(st, x)$ *which outputs* ak, ch*.*
- *There is an additional (possibly quantum) efficient algorithm* $\mathsf{MallPunc}(st, x)$ *which outputs* C_{punc} *such that for any* x'*: if* $Q_{\mathsf{rel}}(x') \neq x$ *then* $C_{\mathsf{punc}}(x') = C(x)$*, where* $x \leftarrow \mathcal{D}_{\mathsf{inp}}(st)$*.*

Malleable-Puncturable $p^{\mathcal{F}}_{triv}$-Security: *For any QPT adversary* $\mathcal{A}$*, we have* $\Pr[\mathsf{MalleablePuncturingGame}_{\mathcal{A}}(1^\lambda)$ $p^{\mathcal{F}}_{triv}(\lambda) + \mathsf{negl}(\lambda)$ *where the security game* $\mathsf{MalleablePuncturingGame}$ *is defined as follows.*

$\mathsf{MalleablePuncturingGame}_{\mathcal{A}}(1^\lambda)$

1. ***Setup Phase:*** Chal *and* $\mathcal{A}$ *interact,* Chal *outputs* $k = (C, aux, Q_{\mathsf{rel}})$*.*
2. *The challenger runs* $x \leftarrow \mathcal{D}_{\mathsf{inp}}(st)$*.*
3. *The challenger runs* $ak, ch \leftarrow \mathsf{SampChFromInp}(st, x)$*.*
4. *The challenger runs* $C_{\mathsf{punc}} \leftarrow \mathsf{MallPunc}(st, x)$*.*
5. ***Challenge Phase:*** *The adversary* $\mathcal{A}$ *receives* $(C_{\mathsf{punc}}, Q_{\mathsf{rel}}), aux, ch, x$*.*
6. *The challenger outputs* 1 *if* $\mathsf{Ver}(ak, ans') = \mathsf{TRUE}$*, otherwise outputs* 0*.*

Weak Challenge Resampling Correctness: [6] *Any malleable-punctured key* $C_{\mathsf{punc}}, Q_{\mathsf{rel}}, aux$ *can be used to sample from the challenge distribution* $\mathsf{SampCh}(st)$*.*

Finally, we also require that the distribution $\mathcal{D}_{\mathsf{inp}}(st)$ has min-entropy λ^c for some $c \in \mathbb{R}^+$, given the state of the adversary $\mathcal{A}$ and st.

3.2 Quantum Protection Definitions

We recall copy-protection security, introduced by [Aar09], which is also referred to as *anti-piracy security* or *unclonability security*. We use the particular formalization given by [ÇG25].

Definition 5 (Copy-Protection)**.** *A quantum protection scheme* $\mathsf{QuantumProtect}$ *is said to satisfy copy-protection security for a cryptographic scheme* $\mathcal{CS} = (\mathsf{Eval}, \mathsf{Ver}, \mathsf{Chal}, \mathsf{SampCh})$ *(Definition 3) if for any QPT adversary* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1\mathcal{A}_2)$*, we have* $\Pr[\mathsf{AntiPiracyGame}_{\mathsf{QuantumProtect},\mathcal{A}}(1^\lambda) = 1] \leq p^{\mathcal{F}}_{triv}(\lambda) + \mathsf{negl}(\lambda)$ *where the security game* $\mathsf{AntiPiracyGame}_{\mathsf{QuantumProtect},\mathcal{A}}(1^\lambda)$ *is defined as follows.*

$\mathsf{AntiPiracyGame}_{\mathsf{QuantumProtect},\mathcal{A}}(1^\lambda)$

1. *Sample* $pp, \mathsf{R}' \leftarrow \mathsf{QuantumProtect.GenState}(1^\lambda)$*.*
2. $\mathsf{Chal}(1^\lambda)$ *and* $\mathcal{A}(1^\lambda)$ *interact,* Chal *outputs* k *and* st*.*
3. *Sample* $cp \leftarrow \mathsf{QuantumProtect.Protect}(pp, k)$*.*
4. *Submit* cp, R' *to* $\mathcal{A}_0$*.*
5. *The adversary* $\mathcal{A}_0$ *outputs two registers* $\mathsf{R}_1, \mathsf{R}_2$*.*

[6] Note that this is trivially true for public-key primitives

6. *The challenger runs* $ak_1, ch_1 \leftarrow \mathsf{SampCh}(st)$ *and* $ak_2, ch_2 \leftarrow \mathsf{SampCh}(st)$.
7. $\mathcal{A}_1$ *receives* ch_1, R_1, *outputs* ans'_1.
8. $\mathcal{A}_2$ *receives* ch_2, R_2, *outputs* ans'_2.
9. $b_{1,Ver} \leftarrow \mathsf{Ver}(ak_1, ans'_1)$ *and* $b_{2,Ver} \leftarrow \mathsf{Ver}(ak_2, ans'_2)$.
10. *The challenger outputs* 1 *if* $b_{1,Ver} = 1 \wedge b_{2,Ver} = 1$, *otherwise outputs* 0.

We also recall strong anti-piracy, which uses *threshold implementations*. We refer the reader to [ALL+21] and [ÇG24a] for definition of threshold implementations. When we write $\mathsf{TI}_{(\mathsf{Ver},\mathcal{D}),p^{\mathcal{F}}_{triv}+\gamma}$ for some distribution $\mathcal{D}$ (such as $\mathsf{SampCh}(st)$), we mean the threshold implementation for the following mixture of binary projective measurements: Sample $ak, ch \leftarrow \mathcal{D}$ and apply the universal quantum circuit to execute the encoded circuit in R on input ch to obtain an outcome ans', and run $\mathsf{Ver}(ak, ans')$ to obtain a bit. When it is clear from context, we will omit writing Ver and just denote the challenge distribution.

Definition 6 (Strong Anti-Piracy)**.** *Let* γ *be an inverse polynomial. A quantum protection scheme* $\mathsf{QuantumProtect}$ *is said to satisfy strong copy-protection security if for any QPT adversary* $\mathcal{A} = \mathcal{A}_0$, *we have* $\Pr\left[\mathsf{StrongAntiPiracyGame}^{\gamma(\lambda)}_{\mathsf{QuantumProtect},\mathcal{A}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$ *where the security game* $\mathsf{StrongAntiPiracyGame}^{\gamma(\lambda)}_{\mathsf{QuantumProtect},\mathcal{A}}(1^\lambda)$ *is defined as follows.*

$\mathsf{StrongAntiPiracyGame}^{\gamma(\lambda)}_{\mathsf{QuantumProtect},\mathcal{A}}(1^\lambda)$

1. *Sample* $pp, \mathsf{R}' \leftarrow \mathsf{QuantumProtect.GenState}(1^\lambda)$.
2. $\mathsf{Chal}(1^\lambda)$ *and* $\mathcal{A}(1^\lambda)$ *interact,* Chal *outputs* k *and* st.
3. *Sample* $cp \leftarrow \mathsf{QuantumProtect.Protect}(pp, k)$.
4. *Submit* cp, R' *to* $\mathcal{A}_0$.
5. *The adversary* $\mathcal{A}_0$ *outputs two registers* $\mathsf{R}_1, \mathsf{R}_2$.
6. *Apply the threshold implementation* $\mathsf{TI}_{(\mathsf{Ver},\mathsf{SampCh}(st)),p^{\mathcal{F}}_{triv}+\gamma} \otimes \mathsf{TI}_{(\mathsf{Ver},\mathsf{SampCh}(st)),p^{\mathcal{F}}_{triv}+\gamma}$ *to* $\mathsf{R}_1, \mathsf{R}_2$.
7. *The challenger outputs* 1 *if both measurements output* 1, *otherwise outputs* 0.

As shown by [CLLZ21], a scheme that satisfies strong anti-piracy for any inverse polynomial γ also satisfies regular anti-piracy (Definition 5).

4 Technical Tools

In this section, we recall some technical tools, mostly verbatim from [ÇG25].

4.1 Quantum Protection Properties of Coset States

We first give the following lemma, which essentially helps separate the computational and the information-theoretic quantum protection properties of coset states.

Lemma 2. *Consider the following (parametrized) game* $\mathsf{Game}^b_{\mathcal{A}}(1^\lambda)$ *(for* $c \in \{0,1\}$*) between a challenger and an adversary* $\mathcal{A}$.

$\underline{\mathsf{Game}^c_{\mathcal{A}}(1^\lambda)}$

1. *The challenger samples a $\mathbb{F}_2^{d(\lambda)}$-coset as follows: Sample a subspace $A \leq \mathbb{F}_2^{d(\lambda)}$ and two elements $a_1, a_2 \in \mathbb{F}_2^{d(\lambda)}$ uniformly at random.*
2. *The challenger initialize the register R with the coset state $|A_{a_1,a_2}\rangle = \sum_{v\in A}(-1)^{\langle v,a_2\rangle}|v+a_1\rangle$.*
3. *The challenger samples a superspace B_1 of A of dimension $\frac{3d}{4}$, and a subspace B_2 of A of dimension $\frac{d}{4}$ and elements $z_1 \leftarrow B_1$, $z_2 \leftarrow B_2^\perp$. The challenger also sets $t = z_1 + a_1$ and $t' = z_2 + a_2$.*
4. *The challenger samples $\hat{M} \leftarrow i\mathcal{O}(M^c)$.*

> $\underline{M^0(b,v)}$
> **Hardcoded:** A, a_1, a_2
>
> 1. *If $b = 0$: Check if $v \in A + a_1$. If so, output* TRUE*, otherwise output* FALSE*, and terminate.*
> 2. *If $b = 1$: Check if $v \in A^\perp + a_2$. If so, output* TRUE*, otherwise output* FALSE*.*

> $\underline{M^1(b,v)}$
> **Hardcoded:** t, t', B_1, B_2
>
> 1. *If $b = 0$: Check if $v \in B_1 + t$. If so, output* TRUE*, otherwise output* FALSE*, and terminate.*
> 2. *If $b = 1$: Check if $v \in B_2^\perp + t'$. If so, output* TRUE*, otherwise output* FALSE*.*

5. *The challenger sends $\mathsf{R}, A, B_1, B_2, t, t'$ to the adversary $\mathcal{A}$.*
6. *The adversary outputs a register $\mathsf{R}_{\mathsf{out}}$.*
7. *The challenger outputs $\mathsf{R}_{\mathsf{out}}, A, a_1, a_2$.*

For any QPT adversary $\mathcal{A}$, $\left\|\mathsf{Game}^0_{\mathcal{A}}(1^\lambda) - \mathsf{Game}^1_{\mathcal{A}}(1^\lambda)\right\|_{Tr} \leq \mathsf{negl}(\lambda)$, assuming the existence of indistinguishability obfuscation and injective one-way functions.

Assuming the existence of subexponentially secure indistinguishability obfuscation and injective one-way functions, there exists a constant[7] $c > 0$ such that that for any quantum adversary that runs in time $2^{-(d(\lambda))^c}$, $\left\|\mathsf{Game}^0_{\mathcal{A}}(1^\lambda) - \mathsf{Game}^1_{\mathcal{A}}(1^\lambda)\right\|_{Tr} \leq 2^{-(d(\lambda))^c}$.

Now we recall the following information-theoretic copy-protection security for coset states.

Lemma 3 (Monogamy-of-Entanglement for Coset States ([CLLZ21], implicit))**.** *Consider the following game between a tuple of adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ and a challenger.*

[7] Note that this is a fixed constant that cannot be improved by assuming higher levels of security for iO or OWF.

$\underline{\mathsf{Game}_{\mathcal{A}}(1^\lambda)}$

1. *The challenger samples a* $\mathbb{F}_2^{d(\lambda)}$*-coset as follows: Sample a subspace* $A \leq \mathbb{F}_2^{d(\lambda)}$ *and two elements* $a_1, a_2 \in \mathbb{F}_2^{d(\lambda)}$ *uniformly at random.*
2. *The challenger initialize the register* R *with the coset state* $|A_{a_1,a_2}\rangle = \sum_{v\in A}(-1)^{\langle v,a_2\rangle}|v+a_1\rangle$*.*
3. *The challenger samples a superspace* B_1 *of* A *of dimension* $\frac{3d}{4}$*, and a subspace* B_2 *of* A *of dimension* $\frac{d}{4}$ *and elements* $z_1 \leftarrow B_1$*,* $z_2 \leftarrow B_2^\perp$*. The challenger also sets* $t = z_1 + a_1$ *and* $t' = z_2 + a_2$*.*
4. *The challenger sends* $\mathsf{R}, B_1, B_2, t, t'$ *to the adversary* $\mathcal{A}_0$*.*
5. *The adversary* $\mathcal{A}_0$ *outputs two registers* $\mathsf{R}_1, \mathsf{R}_2$*.*
6. *The adversaries*[8] $\mathcal{A}_1, \mathcal{A}_2$ *receive* (A, R_1) *and* (A, R_2)*, respectively.*
7. *The adversaries* $\mathcal{A}_1, \mathcal{A}_2$ *outputs* v *and* w*, respectively.*
8. *The challenger checks if* $v = \mathsf{Can}(A, a_1)$ *and* $w = \mathsf{Can}(A^\perp, a_2)$*. If so, it outputs* 1*. Otherwise, it outputs* 0*.*

For any (unbounded) adversary $\mathcal{A}$*,*

$$\Pr\left[\mathsf{Game}_{\mathcal{A}}(1^\lambda) = 1\right] \leq 2^{-c_{MoE}\cdot d(\lambda)}$$

4.2 Quantum-Proof Reconstructive Extractors

We recall the notion of *quantum-proof* extractors, which is a randomness extractor that is secure against quantum side information on the source.

Definition 7 (Quantum-proof seeded extractor [DPVR12])**.** *A function* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is said to be a* (k,ϵ)-strong quantum-proof seeded extractor *if for any cq-state* $\rho \in \{0,1\}^n \otimes \mathcal{Y}$ *of the registers* X, Y *with* $H_{\mathsf{min}}(X|Y) \geq k$*, we have*

$$\mathsf{Ext}(X,S), Y, S \approx_\epsilon U_m, Y, S$$

where $S \leftarrow \{0,1\}^d$ *and* $U_m \leftarrow \{0,1\}^m$*.*

We call an extractor *reconstructive extractor* if given a distinguisher between $\mathsf{Ext}(X,S), Y, S$ versus U_m, Y, S with advantage better than ϵ, there exists an (unbounded) algorithm $\mathcal{E}$ such that $\Pr[\mathcal{E}(Y) = X] \geq 2^{-k}$.

4.3 Projective and Threshold Implementations

We recall the following properties of projective and threshold implementations ([Zha20, ALL+21]). We refer the reader to [ALL+21] and [ÇG24a] for definitions of projective and threshold implementations.

[8] The adversaries not communicating, but they are entangled through $\mathsf{R}_1, \mathsf{R}_2$

Theorem 5 ([ALL+21]). *Consider the approximate threshold implementation $\mathsf{ATI}^{\epsilon,\delta}_{(\mathcal{P},\mathcal{D}),\eta}$, associated with a collection of binary 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.*

We then have the following.

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

We give some further generalizations below.

Theorem 6 ([ÇG24a]). *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 $\mathrm{Tr}\Big[\Big(\bigotimes_{\ell\in[k]} \mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_\ell,\mathcal{D}_\ell,\eta_\ell}\Big) \cdot \rho\Big]$ to denote the probability that the outcome of the joint measurement $\bigotimes_{\ell\in[k]} \mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_\ell,\mathcal{D}_\ell,\eta_\ell}$ applied on ρ is all 1, and similarly for TI.*

Then, we have the following.

- *For any k-partite state ρ,*
$$\mathrm{Tr}\left[\left(\bigotimes_{\ell\in[k]} \mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_\ell,\mathcal{D}_\ell,\eta_\ell-\epsilon}\right)\rho\right] \geq \mathrm{Tr}\left[\left(\bigotimes_{\ell\in[k]} \mathsf{TI}_{\eta_\ell}(\mathcal{P}_{\ell\mathcal{D}_\ell})\right)\rho\right] - k\cdot\delta.$$
- *For any k-partite state ρ, let ρ' be the collapsed state obtained after applying $\bigotimes_{\ell\in[k]} \mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_\ell,\mathcal{D}_\ell,\eta_\ell}$ to ρ and obtaining the outcome 1. Then,*
$$\mathrm{Tr}\left[\left(\bigotimes_{\ell\in[k]} \mathsf{TI}_{\eta_\ell-2\epsilon}(\mathcal{P}_{\ell\mathcal{D}_\ell})\right)\rho'\right] \geq 1 - 2k\cdot\delta.$$
- *For any k-partite state ρ, let ρ' be the collapsed state obtained after applying $\bigotimes_{\ell\in[k]} \mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_\ell,\mathcal{D}_\ell,\eta}$ to ρ and obtaining the outcome 1. Then,*
$$\mathrm{Tr}\left[\left(\bigotimes_{\ell\in[k]} \mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_\ell,\mathcal{D}_\ell,\eta_\ell-3\epsilon}\right)\cdot\rho'\right] \geq 1 - 3k\cdot\delta.$$
- *For any k-partite state ρ,*
$$\mathrm{Tr}\left[\left(\bigotimes_{\ell\in[k]} \mathsf{TI}_{\eta_\ell-\epsilon}(\mathcal{P}_{\ell\mathcal{D}_\ell})\right)\rho\right] \geq \mathrm{Tr}\left[\left(\bigotimes_{\ell\in[k]} \mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_\ell,\mathcal{D}_\ell,\eta_\ell}\right)\rho\right] - k\cdot\delta.$$

We now give the following lemma about simulating approximate threshold implementations.

Lemma 4 (Simulated ATI ([ÇG25]))**.** *There exists a polynomial $q(\cdot)$ and a (parametrized) measurement $\mathsf{SimATI}^{\epsilon,\delta,\alpha}_{P,\eta}$ that receives as input a list of ℓ strings (from the support of $\mathcal{D}$) and runs in time $\mathsf{poly}(\ell)$ such that for any $0 < \epsilon, \delta, \eta, \alpha < 1$, for any collection of binary projective measurements $P = \{P_i, I - P_I\}_{i\in\mathcal{I}}$, for any possibly correlated distribution of states ρ with distributions $\mathcal{D}$ over $\mathcal{I}$,*

$$\Pr\left[\mathsf{SimATI}^{\epsilon,\delta,\alpha}_{P,\eta-5\epsilon}(s_1,\ldots,s_\ell)(\rho) = 1 : s_1,\ldots,s_\ell \leftarrow \mathcal{D}\right] \geq \Pr\left[\mathsf{ATI}^{\epsilon,\delta}_{(P,\mathcal{D}),\eta}(\rho) = 1\right] - \alpha - 4\delta.$$

$$\Pr\left[\mathsf{ATI}^{\epsilon,\delta}_{(P,\mathcal{D}),\eta-5\epsilon}(\rho) = 1\right] \geq \Pr\left[\mathsf{SimATI}^{\epsilon,\delta,\alpha}_{P,\eta}(s_1,\ldots,s_\ell)(\rho) = 1 : s_1,\ldots,s_\ell \leftarrow \mathcal{D}\right] - \alpha - 4\delta$$

$$\Pr\left[\mathsf{SimATI}^{\epsilon,\delta,\alpha}_{P,\eta-5\epsilon}(s_1,\ldots,s_\ell)(\rho) = 1 : s_1,\ldots,s_\ell \leftarrow \mathcal{D}\right] \geq \Pr\left[\mathsf{TI}_{(P,\mathcal{D}),\eta}(\rho) = 1\right] - \alpha - 4\delta$$

$$\Pr\left[\mathsf{TI}_{(P,\mathcal{D}),\eta-5\epsilon}(\rho) = 1\right] \geq \Pr\left[\mathsf{SimATI}^{\epsilon,\delta,\alpha}_{P,\eta}(s_1,\ldots,s_\ell)(\rho) = 1 : s_1,\ldots,s_\ell \leftarrow \mathcal{D}\right] - \alpha - 4\delta$$

where $\ell = q(1/\epsilon, \log(1/\delta), 1/\alpha)$. Note that the runtime of SimATI depends on the parameters of $\mathcal{D}$ (e.g. sampling runtime of $\mathcal{D}$), but otherwise it does not use oracles access to $\mathcal{D}$ nor does it use any samples from $\mathcal{D}$ (except for $s_1,\ldots,s_\ell$ which are given explicitly).

5 Quantum Protection Construction

In this section, we give our quantum protection construction $\mathsf{QuantumProtect}$, which is the same as the construction of [ÇG25], which in turn is essentially the same as the construction of [CLLZ21] (when instantiated with iO and coset states rather than ideal oracles and subspace states).

Let $\mathcal{CS} = (\mathsf{Eval}, \mathsf{Ver}, \mathsf{Chal}, \mathsf{SampCh})$ be the malleable-puncturable scheme to be protected. Let PRF be a pseudorandom function family scheme and let $i\mathcal{O}$ be an indistinguishability obfuscation scheme, with security levels and input-output sizes to be defined in the proof. Let $d(\lambda)$ be a parameter (set in the proof). Further, all the obfuscated programs are implicitly padded to the appropriate sizes as needed in the proof.

$\underline{\mathsf{QuantumProtect.GenState}(1^\lambda)}$

1. Sample a $\mathbb{F}_2^{d(\lambda)}$-coset as follows: Sample a subspace $A \leq \mathbb{F}_2^{d(\lambda)}$ and two elements $a_1, a_2 \in \mathbb{F}_2^{d(\lambda)}$ uniformly at random.
2. Initialize the register R with the coset state $|A_{a_1,a_2}\rangle = \sum_{v\in A}(-1)^{\langle v,a_2\rangle}|v + a_1\rangle$.
3. Sample $\hat{M} \leftarrow i\mathcal{O}(M)$ where M is the membership checking program for the cosets as defined below.

   $\underline{M(b,v)}$

   Hardcoded: A, a_1, a_2

   1. If $b = 0$: Check if $v \in A + a_1$. If so, output TRUE, otherwise output FALSE, and terminate.
   2. If $b = 1$: Check if $v \in A^\perp + a_2$. If so, output TRUE, otherwise output FALSE.

4. Output $(pp = \hat{M}, \mathsf{R})$.

$\underline{\mathsf{QuantumProtect.Protect}(pp, k)}$

1. Parse $\hat{M} = pp$.
2. Parse $(C, aux, Q_{\mathsf{rel}}) = k$.
3. Sample a PRF key as $K \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$.
4. Sample $\hat{P} \leftarrow i\mathcal{O}(P)$ where P is the following program.

   $\underline{P(x, b, v)}$

   Hardcoded: $K, \hat{M}, C$

   1. Check if $\hat{M}(b, v) = \mathsf{TRUE}$. If not, output $\perp$ and terminate.
   2. If $b = 0$, output $C(x) \oplus \mathsf{PRF.Eval}(K, x)$ and terminate.
   3. If $b = 1$, output $\mathsf{PRF.Eval}(K, x)$.

5. Output $(\hat{P}, aux)$.

$\underline{\mathsf{QuantumProtect.Eval}(\mathsf{R}, x)}$

1. Parse $((\hat{P}, aux), \mathsf{R}') = \mathsf{R}$.
2. Run $\mathcal{CS}.\mathsf{Eval}(aux, z)$ by implementing each circuit-query as follows: Run $\hat{P}$ on $x, 0, \mathsf{R}'$ coherently, let y_0 denote the outcome. Then, rewind the register R' (as in Lemma 1). Then run $\hat{P}$ on $x, 1, \mathsf{R}'$ coherently, let y_1 denote the outcome. Then, rewind the register R' (as in Lemma 1). Pass $y_0 \oplus y_1$ as the output.

Theorem 7. *For any malleable-puncturable scheme $\mathcal{CS}$,* QuantumProtect *satisfies strong anti-piracy (Definition 6) for all inverse polynomial $\gamma(\lambda)$.*

Corollary 1. *Assuming subexponentially secure indistinguishability obfuscation and one-way functions,* QuantumProtect *is a secure copy-protection for any malleable-puncturable scheme.*

Proof. All the assumed primitives (in the construction and in the proof) can be instantiated using subexponentially secure indistinguishability obfuscation and one-way functions for any subexponential function. Correctness of the scheme is straightforward. The security follows from Theorem 7 and the fact that strong anti-piracy for all inverse polynomial values implies regular anti-piracy (Definition 5), as shown by [CLLZ21]. □

6 Proof of Copy-Protection Security

In this section, we prove Theorem 7. We prove security through a sequence of hybrids, each of which is constructed by modifying the previous one.

Let $\mathcal{CS} = (\mathsf{Eval}, \mathsf{Ver}, \mathsf{Chal}, \mathcal{D}_{\mathsf{inp}}, \mathsf{SampChFromInp}, \mathsf{MallPunc})$ be the malleable-puncturable scheme being copy-protected. Suppose for a contradiction that there exists a QPT adversary $\mathcal{A}$ and polynomials $q_1(\lambda), q_2(\lambda)$ such that $\Pr\left[\mathsf{StrongAntiPiracyGame}^{\gamma(\lambda)}_{\mathsf{QuantumProtect},\mathcal{A}}(1^\lambda) = 1\right] \geq \frac{1}{q_1(\lambda)}$ for infinitely many values of $\lambda > 0$, where $\gamma(\lambda) = \frac{1}{q_2(\lambda)}$. We also set the parameter $\epsilon^*(\lambda) = \frac{\gamma(\lambda)}{128}$. Another

parameter, $\delta^*(\lambda)$, will be chosen to satisfy the parameter constraints given in the proof.Let Ext be a constructive extractor with output length $\lambda^{c'}$ for some $c' \in \mathbb{R}^+$ (any c' works, the precise value will depend on other parameters). against quantum side-information (Section 4.2) for sources that are 2^{-k} unpredictable sources. Let ϵ_{Ext} denote the extractor error level. Let $p_1(\lambda)$ denote the input size of the circuits of $\mathcal{CS}$, that is, $\mathcal{X} = \{0,1\}^{p_1(\lambda)}$. Let $p_5(\lambda)$ denote the output size of the circuits of $\mathcal{CS}$. Let $p_2(\lambda)$ denote the maximum of the circuit size of $\mathcal{CS}$ and the key length of the PRF scheme PRF. Let $\alpha_2(\lambda)$ denote the security of ACE, let $\alpha_3(\lambda)$ denote iO security, let $\alpha_4(\lambda)$ denote the security level from Lemma 2. $\alpha_1(\lambda)$ is a parameter that will be clear from the proof. Let G be a PRG with appropriate input-output length.

$\underline{\mathsf{Hyb}_0}$**:** The original security game $\mathsf{StrongAntiPiracyGame}^{\gamma(\lambda)}_{\mathsf{QuantumProtect},\mathcal{A}}(1^\lambda)$. As a notation, we write $b^1_{Test,1}$ and $b^1_{Test,2}$ to denote the measurement outcomes obtained by the challenger by applying the threshold implementations to the registers R_1 and R_2, respectively. We set $b^1_{Test} = b^1_{Test,1} \wedge b^1_{Test,2}$. Thus, outcome of the experiment in this hybrid is b^1_{Test}.

$\underline{\mathsf{Hyb}_1}$**:** Instead of applying the threshold implementation $\mathsf{TI}_{\mathsf{SampCh}(st),p^{\mathcal{F}}_{triv}+\gamma(\lambda)}$ on the registers $\mathsf{R}_1, \mathsf{R}_2$, the challenger now instead applies approximate threshold implementation $\mathsf{ATI}^{(1)} = \mathsf{ATI}^{\epsilon^*,\delta^*}_{\mathsf{SampCh}(st),p^{\mathcal{F}}_{triv}+\frac{126\gamma}{128}}$ on the registers R_1 and R_2. We still write $b^1_{Test,1}, b^1_{Test,2}$ to denote the measurement outcomes, respectively. Note that the output of the game is b^1_{Test} in this hybrid.

$\underline{\mathsf{Hyb}_2}$**:** We modify the sampling of $\hat{M}$ during the execution of $\mathsf{QuantumProtect.GenState}$. We first sample a superspace B_1 of A of dimension $\frac{3d}{4}$, a subspace B_2 of A of dimension $\frac{d}{4}$ and elements $u_1 \leftarrow B_1$, $u_2 \leftarrow B_2^\perp$. Finally, we set $t = u_1 + a_1$ and $t' = u_2 + a_2$. Now, we sample $\hat{M} \leftarrow i\mathcal{O}(M')$ where M' is the following program.

$\underline{M'(b,v)}$
Hardcoded: t, t', B_1, B_2

1. If $b = 0$: Check if $v \in B_1 + t$. If so, output TRUE, otherwise output FALSE, and terminate.
2. If $b = 1$: Check if $v \in B_2^\perp + t'$. If so, output TRUE, otherwise output FALSE.

$\underline{\mathsf{Hyb}_3}$**:** After applying $\mathsf{ATI}^{(1)}$ on the registers R_1 and R_2, the challenger applies $\mathsf{TI}^{(2)} \otimes \mathsf{TI}^{(2)}$ to the registers $\mathsf{R}_1, \mathsf{R}_2$ where $\mathsf{TI}_{\mathsf{SampCh}(st),p^{\mathcal{F}}_{triv}+\frac{122\gamma}{128}}$. We write $b^2_{Test,1}, b^2_{Test,2}$ to denote the measurement outcomes, respectively. Finally, the challenger now outputs $b^1_{Test,1}, b^1_{Test,2}, b^2_{Test,1}, b^2_{Test,2}$ as the output of the game.

$\underline{\mathsf{Hyb}_4}$**:** The challenger samples an ACE instance as $sk \leftarrow \mathsf{ACE.Setup}(1^\lambda)$, $ek = \mathsf{ACE.GenEK}(sk, \mathsf{FALSE}^{[9]})$ and $dk' = \mathsf{ACE.GenDK}(sk, \mathsf{TRUE}^{[10]})$. Then it samples $r^*_{\mathsf{ct},1}, r^*_{\mathsf{ct},2} \leftarrow \{0,1\}^{p_4(\lambda)}$ and computes

[9] This represents the circuit that always outputs FALSE, meaning that the encapsulation key will not be punctured at all.

[10] This represents the circuit that always outputs TRUE, meaning that the encapsulation key will be punctured everywhere.

$ct_1^* = C^* \oplus G(r^*_{\mathsf{ct},1}), ct_2^* = K \oplus G(r^*_{\mathsf{ct},2})$ and $a_1^{Can} = \mathsf{Can}(A, a_1), a_2^{Can} = \mathsf{Can}(A^\perp, a_2)$. We also modify the way the challenger samples $\hat{P}$: Now it samples it as $\hat{P} \leftarrow i\mathcal{O}(P^{(1)})$.

$P^{(1)}(x, b, v)$

Hardcoded: $K, \hat{M}, C, Q_{\mathsf{rel}}, dk', ct_1^*, ct_2^*$

1. Check if $\hat{M}(b, v) = \mathsf{TRUE}$. If not, output $\perp$ and terminate.
2. Compute $x' = Q_{\mathsf{rel}}(x)$.
3. Compute $pl = \mathsf{ACE.Dec}(dk', x')$. If $pl \neq \perp$, parse $t||A'||se'||z||ind = pl$.
4. If $pl = \perp$ or if $b = 0 \wedge t = 1$ or if $b = 1 \wedge t = 0$,
    1. If $b = 0$, output $C(x) \oplus \mathsf{PRF.Eval}(K, x)$ and terminate.
    2. If $b = 1$, output $\mathsf{PRF.Eval}(K, x)$ and terminate.
5. If $pl \neq \perp$,
    1. If $t = 0$,
        1. Compute $a' = \mathsf{Can}(A', v)$ and $r' = \mathsf{Ext}(se', a') \oplus z$.
        2. Compute $C' = ct_1^* \oplus G(r')$.
        3. Output $C'(x) \oplus \mathsf{PRF.Eval}(K, x)$ and terminate.
    2. If $t = 1$,
        1. Compute $a' = \mathsf{Can}((A')^\perp, v)$ and $r' = \mathsf{Ext}(se', a') \oplus z$.
        2. Compute $K' = ct_2^* \oplus G(r')$.
        3. Output $\mathsf{PRF.Eval}(K', x)$ and terminate.

Hyb_5: The challenger now sets $dk' = \mathsf{ACE.GenDK}(sk, \mathsf{FALSE})$.

Hyb_6: We define the following distributions where

- $ek_1 = \mathsf{ACE.GenEK}(sk, PRE_1)$ and $PRE_1(m)$ is the circuit that outputs TRUE if the first bit of m is not 1. That is, ek_1 can only encapsulate messages that start with 1.
- $ek_0 = \mathsf{ACE.GenEK}(sk, PRE_0)$ and $PRE_0(m)$ is the circuit that outputs TRUE if the first bit of m is not 0. That is, ek_0 can only encapsulate messages that start with 0.

$\mathcal{D}_1^{(2)}$

Hardcoded: st, A, ek_0

1. Sample an extractor seed as $se \leftarrow \{0,1\}^{\ell_3(\lambda)}$.
2. Sample $r_1^{**} \leftarrow \{0,1\}^{p_4(\lambda)}$.
3. Set $pl_1 = 0||A||se||r_1^{**}||0^{\ell_4(\lambda)}$.
4. Sample $x^* \leftarrow \mathsf{ACE.StegEnc}(ek_0, pl_1, \langle \mathcal{D}_{\mathsf{inp}}(st) \rangle)$.

5. Sample $ak, ch \leftarrow \mathsf{SampChFromInp}(st, x^*)$.

6. Output ak, ch.

$\underline{\mathcal{D}_2^{(2)}}$

Hardcoded: cp^*, C^*, A, ek

1. Sample an extractor seed as $se \leftarrow \{0,1\}^{\ell_3(\lambda)}$.

2. Sample $r_2^{**} \leftarrow \{0,1\}^{p_4(\lambda)}$.

3. Set $pl_2 = 1||A||se||r_2^{**}||0^{\ell_4(\lambda)}$.

4. Sample $x^* \leftarrow \mathsf{ACE.StegEnc}(ek_1, pl_2, \langle \mathcal{D}_{\mathsf{inp}}(st)\rangle)$.

5. Sample $ak, ch \leftarrow \mathsf{SampChFromInp}(st, x^*)$.

6. Output ak, ch.

Finally, instead of applying $\mathsf{TI}^{(2)} \otimes \mathsf{TI}^{(2)}$ to the registers $\mathsf{R}_1, \mathsf{R}_2$, the challenger now applies $\mathsf{ATI}^{(3,1)} \otimes \mathsf{ATI}^{(3,2)}$ to the registers $\mathsf{R}_1, \mathsf{R}_2$, where $\mathsf{ATI}^{(3,1)} = \mathsf{ATI}^{\epsilon^*,\delta^*}_{\mathcal{D}_1^{(2)}, p^{\mathcal{F}}_{triv} + \frac{100\gamma}{128}}$ and $\mathsf{ATI}^{(3,2)} = \mathsf{ATI}^{\epsilon^*,\delta^*}_{\mathcal{D}_2^{(2)}, p^{\mathcal{F}}_{triv} + \frac{100\gamma}{128}}$. We still write $b^2_{Test,1}, b^2_{Test,2}$ to denote the measurement outcomes, respectively. Also, the challenger still outputs $b^1_{Test,1}, b^1_{Test,2}, b^2_{Test,1}, b^2_{Test,2}$ as the output of the game.

$\underline{\mathsf{Hyb}_7}$**:** The challenger flips a random coin $c' \leftarrow \{1,2\}$ and only applies $\mathsf{ATI}^{(3,c')}$ to $\mathsf{R}_{c'}$, instead of testing both registers. Let b^2_{Test} denote the measurement outcome. The challenger now outputs $b^1_{Test,1}, b^1_{Test,2}, b^2_{Test}$ as the experiment outcome.

$\underline{\mathsf{Hyb}_8}$**:** Instead of applying $\mathsf{ATI}^{(3,c')}$ to $\mathsf{R}_{c'}$, the challenger samples $ak^*, ch^* \leftarrow \mathcal{D}^{(2)}_{c'}$ and runs $b^2_{Ver} \leftarrow U(\mathsf{R}_{c'}, ch)$. The challenger now outputs $b^1_{Test,1}, b^1_{Test,2}, b^2_{Ver}$ as the experiment outcome. We write x^*, r^{**}, se^*, pl^* to denote the values sampled/computed during the sampling of ak^*, ch^*.

$\underline{\mathsf{Hyb}_9}$**:** First, we compute z^* as follows. If $c' = 1$, we compute $C' = ct^*_1 \oplus G(\mathsf{Ext}(se^*, a_1^{Can}) \oplus r^{**})$ and $z^* = C'(act^*) \oplus F(K, act^*)$ where C' is interpreted as a circuit. If $c' = 2$, we compute $z^* = \mathsf{PRF.Eval}(K', x^*)$ where $K' = ct^*_2 \oplus G(\mathsf{Ext}(se^*, a_2^{Can}) \oplus r^{**})$. The challenger computes $dk'' = \mathsf{ACE.GenDK}(sk, \mathbb{1}_{pl^*})$.

We modify the way the challenger samples $\hat{P}$: Now it samples it as $\hat{P} \leftarrow i\mathcal{O}(P^{(2)})$.

$\underline{P^{(2)}(x, b, v)}$

Hardcoded: $K, \hat{M}, C, Q_{\mathsf{rel}}, ct^*_1, ct^*_2, dk'', x^*, z^*$

1. Check if $\hat{M}(b, v) = \mathsf{TRUE}$. If not, output $\perp$ and terminate.

2. Compute $x' = Q_{\mathsf{rel}}(x)$.

3. If $x' = x^*$,

    1. If $b = 0$ and $c' = 1$, output z^* and terminate.

2. If $b = 1$ and $c' = 2$, output z^* and terminate.
3. Otherwise if $b = 1$, output $\mathsf{PRF.Eval}(K, x)$ and terminate.
4. Otherwise if $b = 0$, output $C(x) \oplus \mathsf{PRF.Eval}(K, x)$ and terminate.

4. Compute $pl = \mathsf{ACE.Dec}(dk'', x')$. If $pl \neq \bot$, parse $t||A'||se'||z||ind = pl$.
5. If $pl = \bot$ or if $b = 0 \wedge t = 1$ or if $b = 1 \wedge t = 0$,
   1. If $b = 0$, output $C(x) \oplus \mathsf{PRF.Eval}(K, x)$ and terminate.
   2. If $b = 1$, output $\mathsf{PRF.Eval}(K, x)$ and terminate.
6. If $pl \neq \bot$,
   1. If $t = 0$,
      1. Compute $a' = \mathsf{Can}(A', v)$ and $r' = \mathsf{Ext}(se', a') \oplus z$.
      2. Compute $C' = ct_1^* \oplus G(r')$.
      3. Output $C'(x) \oplus \mathsf{PRF.Eval}(K, x)$ and terminate.
   2. If $t = 1$,
      1. Compute $a' = \mathsf{Can}((A')^{\bot}, v)$ and $r' = \mathsf{Ext}(se', a') \oplus z$.
      2. Compute $K' = ct_2^* \oplus G(r')$.
      3. Output $\mathsf{PRF.Eval}(K', x)$ and terminate.

Also, from this hybrid on, our obfuscated programs will have two branches depending on the value of c'. However, since c' is sampled at the beginning of the experiment, we simply (and implicitly) remove the dead branch (i.e. if $c' = 1$, we can remove the lines that are executed if $c' = 1$, along with the hardwired values that are only used there). Note that this does not change the proof thanks to $i\mathcal{O}$ security.

Hyb_{10}: The challenger now samples $x^* \leftarrow \mathcal{D}_{\mathsf{inp}}(st)$ instead of sampling using $\mathsf{ACE.StegEnc}$.

Hyb_{11}: The challenger now samples $\hat{M}$ as $\hat{M} \leftarrow i\mathcal{O}(M)$.

Hyb_{12}: We sample $K\{x^*\} \leftarrow \mathsf{PRF.Punc}(K, x^*)$. We also set $s_1^* = \mathsf{PRF.Eval}(K, x^*)$ and $s_2^* = C(x^*) \oplus s_1^*$. We also sample $C_{\mathsf{punc}} \leftarrow \mathsf{MallPunc}(st, x^*)$. Then, we modify the obfuscated program $\hat{P}$ by now sampling it as $\hat{P} \leftarrow i\mathcal{O}(P^{(3)})$.

$P^{(3)}(x, b, v)$

Hardcoded: $\hat{M}, Q_{\mathsf{rel}}, ct_1^*, ct_2^*, dk'', x^*, z^*, C_{\mathsf{punc}}, K\{x^*\}, s_1^*, s_2^*$

1. Check if $\hat{M}(b, v) = \mathsf{TRUE}$. If not, output $\bot$ and terminate.
2. Compute $x' = Q_{\mathsf{rel}}(x)$.
3. If $x' = x^*$,
   1. If $b = 0$ and $c' = 1$, output z^* and terminate.
   2. If $b = 1$ and $c' = 2$, output z^* and terminate.

3. Otherwise if $b = 1$, output s_1^* and terminate.
4. Otherwise if $b = 0$, output s_2^* and terminate.

4. Compute $pl = \mathsf{ACE.Dec}(dk'', x')$. If $pl \neq \bot$, parse $t||A'||se'||z||ind = pl$.
5. If $pl = \bot$ or if $b = 0 \wedge t = 1$ or if $b = 1 \wedge t = 0$,
   1. If $b = 0$, output $C_{\mathsf{punc}}(x) \oplus \mathsf{PRF.Eval}(K\{x^*\}, x)$ and terminate.
   2. If $b = 1$, output $\mathsf{PRF.Eval}(K\{x^*\}, x)$ and terminate.
6. If $pl \neq \bot$,
   1. If $t = 0$,
      1. Compute $a' = \mathsf{Can}(A', v)$ and $r' = \mathsf{Ext}(se', a') \oplus z$.
      2. Compute $C' = ct_1^* \oplus G(r')$.
      3. Output $C'(x) \oplus \mathsf{PRF.Eval}(K, x)$ and terminate.
   2. If $t = 1$,
      1. Compute $a' = \mathsf{Can}((A')^{\bot}, v)$ and $r' = \mathsf{Ext}(se', a') \oplus z$.
      2. Compute $K' = ct_2^* \oplus G(r')$.
      3. Output $\mathsf{PRF.Eval}(K', x)$ and terminate.

Hyb_{13}**:** We modify the way ct_1^*, ct_2^* are computed: Now we sample $ct_1^*, ct_2^* \leftarrow \{0,1\}^{p_2(\lambda)}$.

Hyb_{14}**:** If $c' = 1$, we sample s_1^* uniformly at random, and also set $z^* = C'(x^*) \oplus s_1^*$ where $C' = ct_1^* \oplus G(\mathsf{Ext}(se^*, a_1^{Can}) \oplus r^{**})$, and also set $s_2^* = \bot$. If $c' = 2$, we sample s_2^* uniformly at random and set $s_1^* = \bot$. Also recall that when $c' = 2$, z^* is computed as $z^* = \mathsf{PRF.Eval}(K', x^*)$ where $K' = ct_2^* \oplus G(\mathsf{Ext}(se^*, a_2^{Can}) \oplus r^{**})$, which does not use the *actual* PRF key K.

Claim 1. *There exists a polynomial $g(\cdot)$ such that $\Pr[b_{Ver}^2 = 1] \geq \frac{1}{g(\lambda)}$ where $b_{Test,1}^1, b_{Test,2}^1, b_{Ver}^2 \leftarrow \mathsf{Hyb}_{14}$.*

We prove this claim in Section 6.1.

Lemma 5. $\Pr[b_{Ver}^2 = 1] \leq \mathsf{negl}(\lambda)$ *where* $b_{Test,1}^1, b_{Test,2}^1, b_{Ver}^2 \leftarrow \mathsf{Hyb}_{14}$.

Proof. This follows directly by the malleable-puncturable security of the scheme $\mathcal{CS}$ since the experiment only uses the punctured circuit C_{punc}. □

Since the statements above contradict each other, our hypothesis that there exists a QPT adversary $\mathcal{A}$ and polynomials $q_1(\lambda), q_2(\lambda)$ such that $\Pr\left[\mathsf{StrongAntiPiracyGame}^{\gamma(\lambda)}_{\mathsf{QuantumProtect},\mathcal{A}}(1^\lambda) = 1\right] \geq \frac{1}{q_1(\lambda)}$ for infinitely many values of $\lambda > 0$ is wrong. Thus, for any polynomial $q_2(\lambda)$, we have $\Pr\left[\mathsf{StrongAntiPiracyGame}^{\gamma(\lambda)}_{\mathsf{QuantumProtect},\mathcal{A}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$ where $\gamma = \frac{1}{q_2(\lambda)}$, which completes the proof.

6.1 Indistinguishability of Hybrids: Proof of Claim 1

Claim 2. $\Pr[b_{Test}^1 = 1] \geq \frac{3}{4q_1(\lambda)}$ *where* $b_{Test}^1 \leftarrow \mathsf{Hyb}_1$.

Proof. Follows from Theorem 6. □

Claim 3. *For* $b^1_{Test,1}, b^1_{Test,2} \leftarrow \mathsf{Hyb}_2$,

- $\Pr\left[b^1_{Test,1} = 1 \wedge b^1_{Test,2} = 1\right] \geq \frac{1}{2q_1(\lambda)}$

Proof. This follows directly from Lemma 2. □

Claim 4. *For* $b^1_{Test,1}, b^1_{Test,2}, b^2_{Test,1}, b^2_{Test,2} \leftarrow \mathsf{Hyb}_3$,

- $\Pr\left[b^1_{Test,1} = 1 \wedge b^1_{Test,2} = 1\right] \geq \frac{1}{2q_1(\lambda)}$
- $\Pr\left[b^2_{Test,1} \wedge b^2_{Test,2} = 1 \middle| b^1_{Test,1} = 1 \wedge b^1_{Test,2} = 1\right] \geq 1 - 6\delta^*$

Proof. Follows from Theorem 6. □

Claim 5. $\mathsf{Hyb}_3 \approx_{\alpha_3} \mathsf{Hyb}_4$

Proof. Observe that P and $P^{(1)}$ have exactly the same functionality: Due to the constrained decapsulation safety of ACE, $\mathsf{ACE.Dec}(dk', x)$ will always output $\perp$ since $dk' = \mathsf{ACE.GenDK}(ek, \mathsf{TRUE})$ (i.e, the key is punctured everywhere). Hence, the result follows by the security of $i\mathcal{O}$. □

Claim 6. $\mathsf{Hyb}_4 \approx_{\alpha_2} \mathsf{Hyb}_5$

Proof. Observe that the experiments do not use the encryption key ek at all, and they do not use any ACE ciphertexts either. Thus, the result follows by the puncture hiding security of ACE. □

Claim 7. *For* $b^1_{Test,1}, b^1_{Test,2}, b^2_{Test,1}, b^2_{Test,2} \leftarrow \mathsf{Hyb}_5$,

- $\Pr\left[b^1_{Test,1} = 1 \wedge b^1_{Test,2} = 1\right] \geq \frac{1}{4q_1(\lambda)}$
- $\Pr\left[b^2_{Test,1} \wedge b^2_{Test,2} = 1 \middle| b^1_{Test,1} = 1 \wedge b^1_{Test,2} = 1\right] \geq 1 - 6\delta^* - 8q_1(\lambda) \cdot (\alpha_3 + \alpha_2)$

Proof. This follows by combining the claims above. □

Claim 8.

- $\Pr\left[b^1_{Test,1} = 1 \wedge b^1_{Test,2} = 1\right] \geq \frac{1}{4q_1(\lambda)}$
- *There exists a polynomial* $g_2(\lambda)$ *such that*

$$\Pr\left[b^2_{Test,1} = 0 \wedge b^2_{Test,2} = 0 \middle| b^1_{Test,1} = 1 \wedge b^1_{Test,2} = 1\right] \leq 1 - \frac{1}{g_2(\lambda)}$$

where $b^1_{Test,1}, b^1_{Test,2}, b^2_{Test,1}, b^2_{Test,2} \leftarrow \mathsf{Hyb}_6$,

Proof. We prove this claim in Section 6.2. □

Claim 9. *For* $b^1_{Test,1}, b^1_{Test,2}, b^2_{Test} \leftarrow \mathsf{Hyb}_7$,

- $\Pr\left[b^2_{Test} = 1 \wedge b^1_{Test,1} = 1 \wedge b^1_{Test,2} = 1\right] \geq \frac{1}{8 \cdot q_1(\lambda) \cdot g_2(\lambda)}$

Proof. This follows directly from the claim above, since with probability $1/2$, c' will *hit* the *good* outcome between $b^2_{Test,1}, b^2_{Test,2}$. □

Claim 10. *For* $b^1_{Test,1}, b^1_{Test,2}, b^2_{Ver} \leftarrow \mathsf{Hyb}_8$,

- $\Pr\left[b^2_{Ver} = 1 \middle| b^1_{Test,1} = 1 \wedge b^1_{Test,2} = 1\right] \geq p^{\mathcal{F}}_{triv} + \frac{90\gamma}{128}$

Proof. This follows easily by the properties of ATI. □

Lemma 6. $\mathsf{Hyb}_8 \approx \mathsf{Hyb}_9$.

Proof. We will prove the two cases $c' = 1$ and $c' = 2$ separately.

First, the case $c' = 1$. We claim that P' and $P^{(2)}$ have exactly the same functionality. Note that once we prove this, the result follows by $i\mathcal{O}$ security. Now we prove our claim. Observe that P' and $P^{(2)}$ can differ in two places: (i) when using the decryption key dk' in the line $pl = \mathsf{ACE.Dec}(dk', x)$, since P' has $dk = \mathsf{ACE.GenDK}(sk, \mathsf{FALSE})$ but $P^{(2)}$ has $dk' = \mathsf{ACE.GenDK}(sk, \mathbb{1}_{pl^*})$, and (ii) if the input satisfies $x = x^*$. First, by the safety of constrained decapsulation of ACE, we know that the output of $\mathsf{ACE.Dec}$ on x can be different for $dk = \mathsf{ACE.GenDK}(sk, \mathsf{FALSE})$ versus $dk' = \mathsf{ACE.GenDK}(sk, \mathbb{1}_{pl^*})$ only if $x = \mathsf{ACE.Enc}(ek, pl^*) = x^*$. Thus, we get that P' and $P^{(2)}$ can possibly differ only on inputs such that $x = x^*$. Further it easy to see that $P'(x, b, v)$ versus $P^{(2)}(x, b, v)$ can only possibly differ if $\hat{M}(b, v) = \mathsf{TRUE}$ is satisfied too. Thus, we only need to consider inputs x, b, v such that $x = act^*$ and $\hat{M}(b, v) = \mathsf{TRUE}$. For such inputs, we will further consider two cases: $b = 0$ and $b = 1$. For $b = 1$, it is easy to see that $P'(x, b, v) = \mathsf{PRF.Eval}(K, x^*) = P^{(2)}(x, b, v)$. Finally, we consider the case $b = 0$. Now, $P'(x, b, v)$ enters the case $pl \neq \perp, t = 0$ since act^* decrypts to pl^* by correctness of decapsulation. Here, it gets $a' = a_1^{Can}$ since $v \in A + a_1$, and then it gets $r' = \mathsf{Ext}(se^*, a_1^{Can}) \oplus r^{**}$ since $pl = pl^*$. Then, it gets $C' = ct_1 \oplus G(\mathsf{Ext}(se^*, a_1^{Can}) \oplus r^{**})$. Thus, we get $P'(x, b, v) = z^*$. Now considering $P^{(2)}(x, b, v)$ when $b = 0$, $x = x^*$ and $v \in A + s$, it is easy to see that we again have $P^{(2,1)}(x, b, v) = z^*$. Thus, $P^{(1)}$ and $P^{(2)}$ have the same output for all x, b, v.

Now we prove the case $c' = 2$. We claim that $P^{(')}$ and $P^{(2)}$ have exactly the same functionality. Similar to above, once we prove this, the result follows by $i\mathcal{O}$ security. The claim that P' and $P^{(2)}$ have the same functionality follows by a case-by-case inspection, similar to above. □

Lemma 7. $\mathsf{Hyb}_9 \approx \mathsf{Hyb}_{10}$.

Proof. Observe that these experiments only use the constrained keys $dk'' = \mathsf{ACE.GenDK}(sk, \mathbb{1}_{pl^*})$ and $ek' = \mathsf{ACE.GenEK}(sk, \mathbb{1}_{pl^*})$. Thus, the result follows by the steganographic ciphertext security of ACE. □

Lemma 8. $\mathsf{Hyb}_{10} \approx \mathsf{Hyb}_{11}$.

Proof. Follows from Lemma 2. □

Lemma 9. $\mathsf{Hyb}_{11} \approx \mathsf{Hyb}_{12}$.

Proof. By the malleable-punctured key correctness of the cryptographic scheme $\mathcal{CS}$ and the punctured key correctness of the scheme PRF, the result follows by the security of $i\mathcal{O}$. □

Lemma 10. $\mathsf{Hyb}_{12} \approx \mathsf{Hyb}_{13}$.

Proof. Observe that the experiments only uses the PRG output $ct^*_1 = G(r^*_1)$ and $ct^*_2 = G(r^*_2)$, and the random strings r^*_1, r^*_2 do not appear anywhere else. Thus, the result follows by the security of G. □

Lemma 11. $\mathsf{Hyb}_{13} \approx \mathsf{Hyb}_{14}$.

Proof. First, observe that the experiments only use the punctured PRF key $K\{act^*\}$. Thus, by the punctured key security of PRF, we can always change $\mathsf{PRF.Eval}(K, act^*)$ to a uniformly random string. This completes the proof of the case $c' = 1$. When $c' = 2$, observe that the experiment does not use $\mathsf{PRF.Eval}(K, act^*)$, thus we can replace $s_2^* = C^*(act^*) \oplus \mathsf{PRF.Eval}(K, act^*)$ with a uniformly random string. □

6.2 Reduction to Monogamy-of-Entanglement: Proof of Claim 8

This part of our proof is mostly the same as [ÇG25].

The first item, that is $\Pr\left[b^1_{Test,1} = 1 \wedge b^1_{Test,2} = 1\right] \geq \frac{1}{4q_1(\lambda)}$, follows easily by the previous claim (since $b^1_{Test,1} = 1, b^1_{Test,2}$ have the same marginal distribution in Hyb_5 and Hyb_4).

Now we prove the second item. Suppose for a contradiction that

$$\Pr\left[b^2_{Test,1} = 0 \wedge b^2_{Test,2} = 0 \middle| b^1_{Test,1} = 1 \wedge b^1_{Test,2} = 1\right] \geq 1 - \mathsf{negl}(\lambda) \tag{1}$$

where $b^1_{Test,1}, b^1_{Test,2}, b^2_{Test,1}, b^2_{Test,2} \leftarrow \mathsf{Hyb}_5$. We will construct an adversary for Lemma 3.

Consider the following modified version Hyb^b_4 of Hyb_4: Simulate Hyb_4 until right after $\mathsf{ATI}^{(1)} \otimes \mathsf{ATI}^{(1)}$ is applied to $(\mathsf{R}_1, \mathsf{R}_2)$, but do not apply $\mathsf{TI}^{(2)} \otimes \mathsf{TI}^{(2)}$. Now, output $\mathsf{R}_1, \mathsf{R}_2, A, ek_b, C^*, cp^*, b^1_{Test,1}, b^1_{Test,2}$ as the outcome of the experiment, where

- $ek_1 = \mathsf{ACE.GenEK}(sk, PRE_1)$ and $PRE_1(m)$ is the circuit that outputs TRUE if the first bit of m is not 1. That is, ek_1 can only encapsulate messages that start with 1.
- $ek_0 = \mathsf{ACE.GenEK}(sk, PRE_0)$ and $PRE_0(m)$ is the circuit that outputs TRUE if the first bit of m is not 0. That is, ek_0 can only encapsulate messages that start with 0.

We write Hyb'_4 to denote the version that outputs both ek_0 and ek_1.

Claim 11. *Let $\mathcal{M}$ be $\mathsf{ATI}^{(3,2)} = \mathsf{ATI}^{\epsilon^*,\delta^*}_{\mathcal{D}^{(2)}_2, p^{\mathcal{F}}_{triv} + \frac{100\gamma}{128}}$. Then,*

$$\Pr\left[\mathsf{TI}^{(2)}(\mathsf{R}_1) = 1 \middle| \mathcal{M}(\mathsf{R}_2, C^*, cp^*, ek_1) = 0 \wedge b^1_{Test,1} = 1 \wedge b^1_{Test,2} = 1\right] \geq 1 - 3 \cdot \sqrt{6\delta^* + 8q_1(\lambda) \cdot (\alpha_3 + \alpha_2)}$$

where $\mathsf{R}_1, \mathsf{R}_2, A, ek_0, ek_1, C^*, cp^*, b^1_{Test,1}, b^1_{Test,2} \leftarrow \mathsf{Hyb}'_4$,

Proof. We already know

$$\Pr\left[(\mathsf{TI}^{(2)} \otimes \mathsf{TI}^{(2)})(\mathsf{R}_1, \mathsf{R}_2) = (1,1) \middle| b^1_{Test,1} = 1 \wedge b^1_{Test,2} = 1\right] \geq 1 - 6\delta^* - 8q_1(\lambda) \cdot (\alpha_3 + \alpha_2).$$

Then the result follows by Equation (1) and Theorem 3 since $\mathsf{TI}^{(2)}$ is a projective measurement. □

Claim 12. *Let $\mathcal{M}$ be $\mathsf{ATI}^{(3,2)} = \mathsf{ATI}^{\epsilon^*,\delta^*}_{\mathcal{D}^{(2)}_2, p^{\mathcal{F}}_{triv} + \frac{100\gamma}{128}}$. Then,*

$$\Pr\left[\mathsf{ATI}^{(3,1)}(\mathsf{R}_1) = 0 \middle| \mathcal{M}(\mathsf{R}_2, C^*, cp^*, ek_1) = 0 \wedge b^1_{Test,1} = 1 \wedge b^1_{Test,2} = 1\right] \geq 1 - \mathsf{negl}(\lambda)$$

where $\mathsf{R}_1, \mathsf{R}_2, A, ek_0, ek_1, C^*, cp^*, b^1_{Test,1}, b^1_{Test,2} \leftarrow \mathsf{Hyb}'_4$.

Claim 13. *Let $\mathcal{M}$ be a QPT algorithm with 1-bit output such that*

- $\Pr[\mathcal{M}(\mathsf{R}_2, C^*, cp^*, ek_1) = 0] \geq \frac{1}{g_3(\lambda)}$ *for some polynomial $g_3(\cdot)$*

- $\Pr\Big[\mathsf{TI}^{(2)}(\mathsf{R}_1)=1\big|\mathcal{M}(\mathsf{R}_2,C^*,cp^*,ek_1)=0\wedge b^1_{Test,1}=1\wedge b^1_{Test,2}=1\Big]\geq 1-3\cdot\sqrt{+6\delta^*+8q_1(\lambda)\cdot(\alpha_3+\alpha_2)},$
- $\Pr\Big[\mathsf{ATI}^{(3,1)}(\mathsf{R}_1)=0\big|\mathcal{M}(\mathsf{R}_2,C^*,cp^*,ek_1)=0\wedge b^1_{Test,1}=1\wedge b^1_{Test,2}=1\Big]\geq 1-\mathsf{negl}(\lambda).$

Then, there exists an (unbounded) algorithm $\mathcal{E}_1^*$ such that

$$\Pr\big[\mathcal{E}_1^*(\mathsf{R}_1,A,ek_0,C^*,cp^*)=a_1^{Can}\big|b^1_{Test,1}=1\wedge b^1_{Test,2}=1\wedge\mathcal{M}(\mathsf{R}_2)=0\big]\geq 2^{-k}$$

where $\mathsf{R}_1,\mathsf{R}_2,A,ek_0,ek_1,C^*,cp^*,b^1_{Test,1},b^1_{Test,2}\leftarrow\mathsf{Hyb}'_4$.

We prove this claim in Section 6.2.1.

Claim 14. *Let $\mathcal{EV}_1$ be an algorithm with 1-bit output such that*

- *$\mathcal{EV}_1$ runs in subexponential time $rt(\lambda)$ (precise value will be clear from the proof)*
- $\Pr[\mathcal{EV}_1(\mathsf{R}_1,A,ek_0,C^*,cp^*)=1]\geq 2^{-k}$
- $$\Pr\Big[\mathsf{TI}^{(2)}(\mathsf{R}_2)=1\big|\mathcal{EV}_1(\mathsf{R}_1,A,ek_0,C^*,cp^*,a_1^{Can})=1\wedge b^1_{Test,1}=1\wedge b^1_{Test,2}=1\Big]\geq$$
  $$1-\frac{3}{2}\cdot\sqrt{6\delta^*+8q_1(\lambda)\cdot(\alpha_3+\alpha_2)}\cdot\frac{2}{2^{-k}},$$
- $\Pr\Big[\mathsf{ATI}^{(3,2)}(\mathsf{R}_2)=0\big|\mathcal{EV}_1(\mathsf{R}_1,A,ek_0,C^*,cp^*,a_1^{Can})=1\wedge b^1_{Test,1}=1\wedge b^1_{Test,2}=1\Big]\geq\frac{1}{4}2^{-k}$

Then, there exists an (unbounded) algorithm $\mathcal{E}_2^*$ such that

$$\Pr\big[\mathcal{E}_2^*(\mathsf{R}_2,A,ek_1,C^*,cp^*)=a_2^{Can}\big|b^1_{Test,1}=1\wedge b^1_{Test,2}=1\wedge\mathcal{EV}_1(\mathsf{R}_1,A,ek_0,C^*,cp^*)=1\big]\geq 2^{-k}$$

where $\mathsf{R}_1,\mathsf{R}_2,A,ek_0,ek_1,C^*,cp^*,b^1_{Test,1},b^1_{Test,2}\leftarrow\mathsf{Hyb}'_4$.

We prove this claim in Section 6.2.1.

Claim 15. *There exists unbounded algorithms $\mathcal{E}_1^*,\mathcal{E}_2^*$ such that*

$$\Pr\big[\mathcal{E}_1^*(\mathsf{R}_1,A,ek_0,C^*,cp^*)=a_1^{Can}\wedge\mathcal{E}_2^*(\mathsf{R}_2,A,ek_1,C^*,cp^*)=a_2^{Can}\big|b^1_{Test,1}=1\wedge b^2_{Test,2}=1\big]\geq(2^{-k})^2\cdot\frac{1}{2}.$$

where $\mathsf{R}_1,\mathsf{R}_2,A,ek_0,ek_1,C^*,cp^*,b^1_{Test,1},b^1_{Test,2}\leftarrow\mathsf{Hyb}'_4$.

Proof. Set $\mathcal{M}=\mathsf{ATI}^{(3,2)}$. First, by Claim 11,Claim 12, and Claim 13, we have that there exist $\mathcal{E}_1^*$ such that

$$p_1=\Pr\big[\mathcal{E}_1^*(\mathsf{R}_1,A,ek_0,C^*,cp^*)=a_1^{Can}\big|b^1_{Test,1}=1\wedge b^1_{Test,2}=1\wedge\mathcal{M}(\mathsf{R}_2,C^*,cp^*,ek_1)=0\big]\geq 2^{-k}$$

which also implies

$$p_1=\Pr\big[\mathcal{E}_1^*(\mathsf{R}_1,A,ek_0,C^*,cp^*)=a_1^{Can}\big|b^1_{Test,1}=1\wedge b^1_{Test,2}=1\big]\geq 2^{-k}\cdot\frac{1}{2}$$

Now, let $\mathcal{EV}_1(1^\lambda)$ be an algorithm simply simulates the output distribution of the equality check $\mathcal{E}_1^*(\mathsf{R}_1,A,ek_0,C^*,cp^*)\stackrel{?}{=}a_1^{Can}$. That is, $\mathcal{EV}_1(1^\lambda)$ simulates a biased coin and outputs 1 with probability p_1. Now we claim the following.

- $$\Pr\Big[\mathsf{TI}^{(2)}(\mathsf{R}_2)=1\Big|\mathcal{EV}(\mathsf{R}_1,A,ek_0,C^*,cp^*,a_1^{Can})=1\wedge b^1_{Test,1}=1\wedge b^1_{Test,2}=1\Big]\geq 1-\frac{3}{2}\cdot\sqrt{6\delta^*+8q_1(\lambda)\cdot(\alpha_3+\alpha_2)}\cdot\frac{2}{2^{-k}},$$

- $\Pr\Big[\mathcal{M}(\mathsf{R}_2,C^*,cp^*,ek_1)=0\Big|\mathcal{EV}(\mathsf{R}_1,A,ek_0,C^*,cp^*,a_1^{Can})=1\wedge b^1_{Test,1}=1\wedge b^1_{Test,2}=1\Big]\geq\frac{1}{4}2^{-k}$

For the first claim, we already know

$$\Pr\Big[(\mathsf{TI}^{(2)}\otimes\mathsf{TI}^{(2)})(\mathsf{R}_1,\mathsf{R}_2)=(1,1)\Big|b^1_{Test,1}=1\wedge b^1_{Test,2}=1\Big]\geq 1-6\delta^*-8q_1(\lambda)\cdot(\alpha_3+\alpha_2).$$

Then the first claim follows by Theorem 3 since $\Pr\Big[\mathcal{EV}((\mathsf{R}_1,A,ek_0,C^*,cp^*),a_1^{Can})=1\Big|b^1_{Test,1}=1\wedge b^1_{Test,2}=1\Big]\geq 2^{-k}$ and $\mathsf{TI}^{(2)}$ is a projective measurement.

To see the second claim, first we have by Bayes' Theorem that,

$$\frac{\Pr\Big[\mathcal{M}(\mathsf{R}_2,C^*,cp^*,ek_1)=0\Big|\mathcal{EV}(\mathsf{R}_1,A,ek_0,C^*,cp^*,a_1^{Can})=1\wedge b^1_{Test,1}=1\wedge b^1_{Test,2}=1\Big]}{\Pr\Big[\mathcal{M}(\mathsf{R}_2,C^*,cp^*,ek_1)=0=0\Big|\wedge b^1_{Test,1}=1\wedge b^1_{Test,2}=1\Big]}\geq$$
$$\Pr\big[\mathcal{EV}(\mathsf{R}_1,A,ek_0,C^*,cp^*,a_1^{Can})=1\big|\mathcal{M}(\mathsf{R}_2,C^*,cp^*,ek_1)=0=0\wedge b^1_{Test,1}=1\wedge b^1_{Test,2}=1\big]$$

The result then follows, since $\Pr\Big[\mathcal{EV}((\mathsf{R}_1,A,ek_0,C^*,cp^*),a_1^{Can})=1\Big|b^1_{Test,1}=1\wedge b^1_{Test,2}=1\Big]\geq\frac{1}{2}2^{-k}$ by above and $\Pr\Big[\mathcal{M}(\mathsf{R}_2,C^*,cp^*,ek_1)=0=0\Big|\wedge b^1_{Test,1}=1\wedge b^1_{Test,2}=1\Big]\geq\frac{1}{2}$ by Equation (1).

Then, applying Claim 14 with $\mathcal{EV}_1(1^\lambda)$, we get

$$\Pr\big[\mathcal{E}_1^*(\mathsf{R}_1,A,ek_0,C^*,cp^*)=a_1^{Can}\wedge\mathcal{E}_2^*(\mathsf{R}_2,A,ek_1,C^*,cp^*)=a_2^{Can}\big|b^1_{Test,1}=1\wedge b^1_{Test,2}=1\wedge\mathcal{M}(\mathsf{R}_2)=0\big]\geq(2^{-k})^2$$

which completes the proof. □

Since $\Pr\Big[b^1_{Test,1}=1\wedge b^2_{Test,2}=1\Big]\geq\frac{1}{4q_1(\lambda)}$, the claim above is a contradiction to Lemma 3 by our choice of parameters. Thus, our contradiction hypothesis that

$$\Pr\big[b^2_{Test,1}=0\wedge b^2_{Test,2}=0\big|b^1_{Test,1}=1\wedge b^1_{Test,2}=1\big]\geq 1-\mathsf{negl}(\lambda),$$

where $b^1_{Test,1},b^1_{Test,2},b^2_{Test,1},b^2_{Test,2}\leftarrow\mathsf{Hyb}_5$, is incorrect. This proves Claim 8.

6.2.1 Proof of Claim 13

We will instead prove the following stronger claim, so that the proof can serve as a proof for both[11] Claim 13 and Claim 14.

Claim 16. *Let $\mathcal{M}$ be an algorithm with 1-bit output such that*

- *$\mathcal{M}$ runs in subexponential time $rt(\lambda)$ (precise value will be clear from the proof)*
- $\Pr[\mathcal{M}(\mathsf{R}_2,C^*,cp^*,ek_1)=0]\geq 2^{-k}$,

[11] Note that while technically Claim 14 is not a consequence of our stronger claim we prove in this section, proof of Claim 14 follows almost exactly the same as the proof of this stronger version of Claim 13.

- $\Pr\left[\mathsf{TI}^{(2)}(\mathsf{R}_1)=1 \middle| \mathcal{M}(\mathsf{R}_2, C^*, cp^*, ek_1)=0 \wedge b^1_{Test,1}=1 \wedge b^1_{Test,2}=1\right] \geq 1-\frac{3}{2}\cdot\sqrt{6\delta^*+8q_1(\lambda)\cdot(\alpha_3+\alpha_2)}\cdot \frac{2}{2^{-k}}$,
- $\Pr\left[\mathsf{ATI}^{(3,1)}(\mathsf{R}_1)=0 \middle| \mathcal{M}(\mathsf{R}_2, C^*, cp^*, ek_1)=0 \wedge b^1_{Test,1}=1 \wedge b^1_{Test,2}=1\right] \geq \frac{1}{4}2^{-k}$.

Then, there exists an (unbounded) algorithm $\mathcal{E}^*_1$ such that

$$\Pr\left[\mathcal{E}^*_1(\mathsf{R}_1, A, ek_0, C^*, cp^*)=a^{Can}_1 \middle| b^1_{Test,1}=1 \wedge b^1_{Test,2}=1 \wedge \mathcal{M}(\mathsf{R}_2)=0\right] \geq 2^{-k}$$

where $\mathsf{R}_1, \mathsf{R}_2, A, ek_0, ek_1, C^*, cp^*, b^1_{Test,1}, b^1_{Test,2} \leftarrow \mathsf{Hyb}'_4$.

We now prove this claim. Let $\mathcal{M}$ be an algorithm as in the claim statement. Define $\mathsf{SimATI}^{(4)} = \mathsf{SimATI}^{\epsilon^*,\delta^*,\alpha_1(\lambda)}_{\frac{116\gamma}{128}}$ and let $t_1(\lambda)$ be the number of samples for $\mathsf{SimATI}^{(4)}$ as in Lemma 4. Then, by Lemma 4, we get

- $\Pr\left[\mathsf{SimATI}^{(4)}(s_{1,1},\ldots,s_{1,t_1(\lambda)})(\mathsf{R}_1)=1 \middle| \mathcal{M}(\mathsf{R}_2, C^*, cp^*, ek_1)=0 \wedge b^1_{Test,1}=1 \wedge b^1_{Test,2}=1\right] \geq 1-\frac{3}{2}\cdot\sqrt{6\delta^*+8q_1(\lambda)\cdot(\alpha_3+\alpha_2)}\cdot\frac{2}{2^{-k}}-\alpha_1-4\delta^*=\xi_1$,
- $\Pr\left[\mathsf{SimATI}^{(4)}(s_{2,1},\ldots,s_{2,t_1(\lambda)})(\mathsf{R}_1)=0 \middle| \mathcal{M}(\mathsf{R}_2, C^*, cp^*, ek_1)=0 \wedge b^1_{Test,1}=1 \wedge b^1_{Test,2}=1\right] \geq \frac{1}{4}2^{-k}-\alpha_1-4\delta^*$.

where $s_{1,1},\ldots,s_{1,t_1(\lambda)} \leftarrow \mathsf{SampCh}(st)$ and $s_{2,1},\ldots,s_{2,t_1(\lambda)} \leftarrow \mathcal{D}^{(1)}_1$.

Now define the following distributions for $j \in [t_1(\lambda)]$.

$\underline{\mathcal{D}^{(2)}_1}$

Hardcoded: st, A, ek_0

1. Sample an extractor seed as $se \leftarrow \{0,1\}^{\ell_3(\lambda)}$.
2. Sample $r^{**}_1 \leftarrow \{0,1\}^{p_4(\lambda)}$.
3. Set $pl_1 = 0||A||se||r^{**}_1||j$.
4. Sample $x^* \leftarrow \mathsf{ACE.StegEnc}(ek_0, pl_1, \langle \mathcal{D}_{\mathsf{inp}}(st)\rangle)$.
5. Sample $ak, ch \leftarrow \mathsf{SampChFromInp}(st, x^*)$.
6. Output ak, ch.

Now we claim the following.

Claim 17. $\Pr\left[\mathsf{SimATI}^{(4)}(s_{2,1},\ldots,s_{2,t_1(\lambda)})(\mathsf{R}_1)=0 \middle| \mathcal{M}(\mathsf{R}_2, C^*, cp^*, ek_1)=0 \wedge b^1_{Test,1}=1 \wedge b^1_{Test,2}=1\right] \geq \frac{1}{4}2^{-k}-\alpha_1-4\delta^*-((\alpha_2+2\alpha_3)\cdot t_1(\lambda)-2\alpha_3)\cdot\frac{1}{2^{-k}}=\xi_2$ *where* $s_{2,j} \leftarrow \mathcal{D}^{(2,j)}_1$ *for* $j \in [t_1(\lambda)]$

Proof. This claim follows easily by an ACE key puncturing and iO argument, changing each sample one by one, since the program ignores the *ind* part of the decrypted messages. □

By applying the hybrid lemma to above, we get that there exists some index $i^* \in \{1,\ldots,t_1(\lambda)\}$ and values τ_1, τ_2 such that

- $\Pr\left[\mathsf{SimATI}^{(4)}((sl_i)_{i\in[t(\lambda)]})(\mathsf{R}_1)=1 \middle| \mathcal{M}(\mathsf{R}_2, C^*, cp^*, ek_1)=0 \wedge b^1_{Test,1}=1 \wedge b^1_{Test,2}=1\right] \geq \tau_1$,

- $\Pr\Big[\mathsf{SimATI}^{(4)}((sr_i)_{i\in[t(\lambda)]})(\mathsf{R}_1)=0\Big|\mathcal{M}(\mathsf{R}_2,C^*,cp^*,ek_1)=0\wedge b^1_{Test,1}=1\wedge b^1_{Test,2}=1\Big]\geq\tau_2$
- $\tau_2-\tau_1\geq\frac{\xi_2-\xi_1}{t_1(\lambda)}$

where

- $\mathsf{R}_1,\mathsf{R}_2,A,ek_0,ek_1,C^*,cp^*,b^1_{Test,1},b^1_{Test,2}\leftarrow\mathsf{Hyb}''_4$ and
- $sl_i\leftarrow\mathcal{D}_1^{(2,i)}$ for $i<i^*$ and $sl_i\leftarrow\mathcal{D}(cp^*,C^*)$ otherwise,
- $sr_i\leftarrow\mathcal{D}_1^{(2,i)}$ for $i\leq i^*$ and $sr_i\leftarrow\mathcal{D}(cp^*,C^*)$ otherwise.

Then we define the following distribution and claim the following.

$\underline{\mathcal{D}_1^{(1,j)}}$

Hardcoded: $cp^*,C^*,A,a_1^{Can},ek,r_1^*$

1. Sample an extractor seed as $se\leftarrow\{0,1\}^{\ell_3(\lambda)}$.
2. Set $pl_1=0||A||se||(\mathsf{Ext}(se,a_1^{Can})\oplus r_1^*)||j$
3. Set $x^*\leftarrow\mathsf{ACE.StegEnc}(ek_0,pl_1)$.
4. Sample $ak,ch\leftarrow\mathsf{SampChFromInp}(st,x^*)$.
5. Output ak,ch.

Claim 18.
$\Pr\Big[\mathsf{SimATI}^{(4)}((sl_i)_{i\in[t(\lambda)]})(\mathsf{R}_1)=1\Big|\mathcal{M}(\mathsf{R}_2,C^*,cp^*,ek_1)=0\wedge b^1_{Test,1}=1\wedge b^1_{Test,2}=1\Big]\geq\tau_1-(2\alpha_2+\alpha_3+2\alpha_4)\cdot\frac{4}{2^{-k}}$ *where*

- $\mathsf{R}_1,\mathsf{R}_2,A,ek_0,ek_1,C^*,cp^*,b^1_{Test,1},b^1_{Test,2}\leftarrow\mathsf{Hyb}'_4$
- $sl_i\leftarrow\mathcal{D}_1^{(2,i)}$ *for* $i<i^*$
- $sl_i\leftarrow\mathcal{D}_1^{(1,i)}$ *if* $i=i^*$,
- $sl_i\leftarrow\mathcal{D}(cp^*,C^*)$ *for* $i>i^*$,

Proof. We will define a sequence of modifications for Hyb'_4.

We define the following modification $\mathsf{Hyb}_4^{'(1)}$ of $\mathsf{Hyb}_4^{'}$: The challenger samples $\hat{M}$ as $\hat{M}\leftarrow i\mathcal{O}(M)$ instead of $\hat{M}\leftarrow i\mathcal{O}(M')$. This essentially undoes the change from Hyb_1 to Hyb_2.

By Lemma 2, we have $\Pr\Big[\mathsf{SimATI}^{(4)}((sl_i)_{i\in[t(\lambda)]})(\mathsf{R}_1)=1\Big|\mathcal{M}(\mathsf{R}_2,C^*,cp^*,ek_1)=0\wedge b^1_{Test,1}=1\wedge b^1_{Test,2}=1\Big]\geq\tau_1-(\alpha_4)\cdot\frac{4}{2^{-k}}$ where

- $\mathsf{R}_1,\mathsf{R}_2,A,ek_0,ek_1,C^*,cp^*,b^1_{Test,1},b^1_{Test,2}\leftarrow\mathsf{Hyb}_4^{'(1)}$
- $sl_i\leftarrow\mathcal{D}_1^{(2,i)}$ for $i<i^*$
- $sl_i\leftarrow\mathcal{D}(cp^*,C^*)$ for $i\geq i^*$,

We define $\mathsf{Hyb}_4^{'(2)}$ so that when the challenger is preparing $\hat{P}$, it now embeds $dk' = \mathsf{ACE.GenDK}(sk, T_{pl^*})$ instead of $dk' = \mathsf{ACE.GenDK}(sk, \mathsf{FALSE})$ where

- $T_{pl^*}(m)$ is the circuit that outputs TRUE if $m = pl^*$,
- $pl^* = \mathsf{ACE.Enc}(ek_0, 0||A||se^*||(\mathsf{Ext}(se^*, a_1^{Can}) \oplus r_1^*)||i^*)$ where se^*, r_1^* are the values sampled during sampling $sl_{i^*} \leftarrow \mathcal{D}_1^{(1,i^*)}$

Since the experiments do not use an encryption of pl^*, we can apply the puncture-hiding security of ACE to get $\Pr\left[\mathsf{SimATI}^{(4)}((sl_i)_{i\in[t(\lambda)]})(\mathsf{R}_1) = 1 \middle| \mathcal{M}(\mathsf{R}_2, C^*, cp^*, ek_1) = 0 \wedge b^1_{Test,1} = 1 \wedge b^1_{Test,2} = 1\right] \geq \tau_1 - (\alpha_2 + \alpha_4) \cdot \frac{4}{2^{-k}}$ where

- $\mathsf{R}_1, \mathsf{R}_2, A, ek_0, ek_1, C^*, cp^*, b^1_{Test,1}, b^1_{Test,2} \leftarrow \mathsf{Hyb}_4^{'(2)}$
- $sl_i \leftarrow \mathcal{D}_1^{(2,i)}$ for $i < i^*$
- $sl_i \leftarrow \mathcal{D}(cp^*, C^*)$ for $i \geq i^*$,

Now, we claim $\Pr\left[\mathsf{SimATI}^{(4)}((sl_i)_{i\in[t(\lambda)]})(\mathsf{R}_1) = 1 \middle| \mathcal{M}(\mathsf{R}_2, C^*, cp^*, ek_1) = 0 \wedge b^1_{Test,1} = 1 \wedge b^1_{Test,2} = 1\right] \geq \tau_1 - (2\alpha_2 + \alpha_4) \cdot \frac{4}{2^{-k}}$ where

- $\mathsf{R}_1, \mathsf{R}_2, A, ek_0, ek_1, C^*, cp^*, b^1_{Test,1}, b^1_{Test,2} \leftarrow \mathsf{Hyb}_4^{'(2)}$
- $sl_i \leftarrow \mathcal{D}_1^{(2,i)}$ for $i < i^*$
- $sl_i \leftarrow \mathcal{D}_1^{(1,i)}$ if $i = i^*$,
- $sl_i \leftarrow \mathcal{D}(cp^*, C^*)$ for $i > i^*$,

This follows by steganographic ciphertext property of ACE since dk' is punctured at pl^*.

We define $\mathsf{Hyb}_4^{'(3)}$ so that when the challenger is preparing $\hat{P}$, it now embeds $dk' = \mathsf{ACE.GenDK}(sk, \mathsf{FALSE})$. We claim $\Pr\left[\mathsf{SimATI}^{(4)}((sl_i)_{i\in[t(\lambda)]})(\mathsf{R}_1) = 1 \middle| \mathcal{M}(\mathsf{R}_2, C^*, cp^*, ek_1) = 0 \wedge b^1_{Test,1} = 1 \wedge b^1_{Test,2} = 1\right] \geq \tau_1 - (\alpha_3 + \alpha_4 + 2\alpha_2) \cdot \frac{4}{2^{-k}}$ where

- $\mathsf{R}_1, \mathsf{R}_2, A, ek_0, ek_1, C^*, cp^*, b^1_{Test,1}, b^1_{Test,2} \leftarrow \mathsf{Hyb}_4^{'(3)}$
- $sl_i \leftarrow \mathcal{D}_1^{(2,i)}$ for $i < i^*$
- $sl_i \leftarrow \mathcal{D}_1^{(1,i)}$ if $i = i^*$,
- $sl_i \leftarrow \mathcal{D}(cp^*, C^*)$ for $i > i^*$,

This follows by the security of $i\mathcal{O}$ since by construction of pl^*, the programs have the same functionality in both cases.

Finally, we claim $\Pr\left[\mathsf{SimATI}^{(4)}((sl_i)_{i\in[t(\lambda)]})(\mathsf{R}_1) = 1 \middle| \mathcal{M}(\mathsf{R}_2, C^*, cp^*, ek_1) = 0 \wedge b^1_{Test,1} = 1 \wedge b^1_{Test,2} = 1\right] \geq \tau_1 - (\alpha_3 + 2\alpha_4 + 2\alpha_2) \cdot \frac{4}{2^{-k}}$ where

- $\mathsf{R}_1, \mathsf{R}_2, A, ek_0, ek_1, C^*, cp^*, b^1_{Test,1}, b^1_{Test,2} \leftarrow \mathsf{Hyb}_4^{'(3)}$
- $sl_i \leftarrow \mathcal{D}_1^{(2,i)}$ for $i < i^*$

- $sl_i \leftarrow \mathcal{D}_1^{(1,i)}$ if $i = i^*$,
- $sl_i \leftarrow \mathcal{D}(cp^*, C^*)$ for $i > i^*$,

This follows directly by Lemma 2. □

The claim above shows that we can distinguish between $\mathcal{D}_1^{(1,i^*)}$ and $\mathcal{D}_1^{(2,i^*)}$ with advantage $\geq \frac{\xi_2-\xi_1}{t_1(\lambda)} - \tau_1 - (2\alpha_2 + \alpha_3 + 2\alpha_4) \cdot \frac{4}{2^{-k}}$. Since this is larger than ϵ_{Ext} by our choice of parameters, we get by the reconstructive property of Ext that there exists an algorithm $\mathcal{E}_1^*$ such that

$$\Pr\left[\mathcal{E}_1^*(\mathsf{R}_1, A, ek_0, C^*, cp^*) = a_1^{Can} \middle| b^1_{Test,1} = 1 \wedge b^1_{Test,2} = 1 \wedge \mathcal{M}(\mathsf{R}_2) = 0\right] \geq 2^{-k}$$

where $\mathsf{R}_1, \mathsf{R}_2, A, ek_0, ek_1, C^*, cp^*, b^1_{Test,1}, b^1_{Test,2} \leftarrow \mathsf{Hyb}'_4$.

This completes the proof.

7 Steganographic Asymmetrically Constrained Encapsulation

In this section, we show how to construct an *asymmetrically constrained encapsulation* (ACE) scheme ([CHJV15]) that satisfies an additional steganographic security property: $\mathsf{ACE.StegEnc}(ek, m, \mathcal{D})$ produces an encapsulation that is indistinguishable from a fresh sample from the distribution $\mathcal{D}$, to any adversary that has keys punctured at m.

7.1 Definition

We first recall the definition of ACE ([CHJV15]). The definition we write below is mostly verbatim from [ÇG25].

Definition 8 (Asymmetrically Constrained Encapsulation)**.** *Let $\mathcal{M} = \{0,1\}^{n(\lambda)}$ denote the message space, where $n(\lambda) = \mathsf{poly}(\lambda)$. Let $\ell(\lambda)$ and $c_{punc}(\lambda)$ be parameters that are polynomials in λ. We define an admissible puncturing circuit to be a circuit of size at most $c_{punc}(\lambda)$ and we define an admissible message to be a message in $\mathcal{M}$. Unless otherwise specified, when we consider a puncturing circuit we will always implicitly consider admissible circuits and we will always consider admissible messages.*

*An **asymmetrically constrained encapsulation scheme** ACE is parametrized by $c_{punc}(\lambda), \ell(\lambda), n(\lambda)$ and consists of the following **efficient** algorithms.*

- $\mathsf{Setup}(1^\lambda)$*: Takes as input the security parameter and outputs a secret key sk.*
- $\mathsf{GenEK}(sk, C)$*: Takes as input the secret key and a (puncturing) circuit C, and outputs a punctured encapsulation key ek.*
- $\mathsf{GenDK}(sk, C)$*: Takes as input the secret key and a circuit C, and outputs a punctured decapsulation key dk.*
- $\mathsf{Enc}(ek, m)$*: Takes as input an encapsulation key ek and a message m, outputs a ciphertext ct or the symbol $\perp$.*
- $\mathsf{Dec}(dk, m)$*: Takes as input a decapsulation key dk and a ciphertext ct, outputs a message m or the symbol $\perp$.*

Note that all of the algorithms, except Setup, are deterministic. Further, the key and ciphertext sizes and the runtimes of the scheme will be a polynomial $n(\lambda), c_{punc}(\lambda), \ell(\lambda)$.

We require the following correctness guarantees and puncture-hiding security (defined later) and pseudorandom ciphertext security (defined later).

Correctness of Decapsulation: *For all (admissible) circuits C, C' and messages m such that $C(m) = C'(m) = \mathsf{FALSE}$,*

$$\Pr\Big[\mathsf{Dec}(\mathsf{GenDK}(sk, C), \mathsf{Enc}(\mathsf{GenEK}(sk, C'), m)) = m : sk \leftarrow \mathsf{Setup}(1^\lambda)\Big] = 1$$

Equivalence of Constrained Encapsulation: *For all circuits C and messages m such that $C(m) = \mathsf{FALSE}$,*

$$\Pr\Big[\mathsf{Enc}(\mathsf{GenEK}(sk, C), m) = \mathsf{Enc}(\mathsf{GenEK}(sk, \mathsf{FALSE}), m) : sk \leftarrow \mathsf{Setup}(1^\lambda)\Big] = 1$$

Here, the notation FALSE *is overloaded to also denote the circuit that outputs* FALSE *everywhere.*

Safety of Constrained Decapsulation: *For all circuits C and strings str,*

$$\Pr\left[\mathsf{Dec}(dk, str) = \perp \vee\ C(\mathsf{Dec}(dk, str)) = \mathsf{FALSE} : \begin{array}{c} sk \leftarrow \mathsf{Setup}(1^\lambda) \\ dk = \mathsf{GenDK}(sk, C) \end{array}\right] = 1$$

Equivalence of Constrained Decapsulation: *For all circuits C and strings str,*

$$\Pr\left[\begin{array}{l} m_1 = m_2 \vee\ C(m_1) = \mathsf{TRUE} : \\ \\ \\ \\ \end{array} \begin{array}{c} sk \leftarrow \mathsf{Setup}(1^\lambda) \\ dk_2 = \mathsf{GenDK}(sk, \mathsf{FALSE}) \\ dk_1 = \mathsf{GenDK}(sk, C) \\ m_2 = \mathsf{Dec}(dk_2, str) \\ m_1 = \mathsf{Dec}(dk_1, str) \end{array}\right] = 1$$

Unique Encapsulations: *There exists a (deterministic) mapping $\mathcal{F}$ from the secret keys and messages to ciphertexts such that*

- *For any $str \neq \mathcal{F}(sk, m)$, $\Pr[\mathsf{Dec}(\mathsf{GenDK}(sk, \mathsf{FALSE}), str) = m] = 0$.*
- $\Pr[\mathsf{Dec}(\mathsf{GenDK}(sk, \mathsf{FALSE}), \mathcal{F}(sk, m)) = m] = 1$.

Now we recall security of constrained decapsulation, also known as puncture-hiding security.

Definition 9 (Security of Constrained Decapsulation (Puncture-Hiding Security))**.** *Consider the following game between a challenger and an adversary.*

$\underline{\mathsf{PunctureHidingGame}_{\mathcal{A}}(1^\lambda)}$

1. *The adversary $\mathcal{A}$ outputs three (admissible) circuits C, C_0, C_1.*
2. *The challenger samples $sk \leftarrow \mathsf{ACE.Setup}(1^\lambda)$ and computes $ek = \mathsf{ACE.GenEK}(sk, C)$, $dk_0 = \mathsf{ACE.GenDK}(sk, C_0)$ and $dk_1 = \mathsf{ACE.GenDK}(sk, C_1)$.*
3. *The challenger samples $b \leftarrow \{0, 1\}$ and sends ek, dk_b to the adversary $\mathcal{A}$.*
4. *The adversary outputs a bit b'.*
5. *The challenger outputs 1 if $b = b'$, and outputs 0 otherwise.*

An asymmetrically constrained encapsulation scheme ACE *is said to satisfy* security of constrained decapsulation *if for any QPT adversary* $\mathcal{A}$*,*

$$\Pr\left[\mathsf{PunctureHidingGame}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Now we recall $\ell(\lambda)$-pseudorandom ciphertext security, introduced by [ÇG25] to generalize $\ell(\lambda)$-ciphertext indistinguishability security requirement originally considered by [CHJV15].

Definition 10 (Pseudorandom Ciphertext Security)**.** *Consider the following game between a challenger and an adversary.*

$\underline{\mathsf{PRCiphertextGame}_{\mathcal{A}}(1^\lambda)}$

1. *The adversary* $\mathcal{A}$ *outputs two (admissible) circuits* C_1, C_2 *and* $\ell(\lambda)$ *messages* $(m_1, \ldots, m_{\ell(\lambda)})$ *such that* $C_1(m_i) = C_2(m_i) = \mathsf{TRUE}$ *for all* $i \in [\ell(\lambda)]$*.*
2. *The challenger samples* $sk \leftarrow \mathsf{ACE.Setup}(1^\lambda)$ *and computes* $ek = \mathsf{ACE.GenEK}(sk, \mathsf{FALSE})$ $ek' = \mathsf{ACE.GenEK}(sk, C_1)$*,* $dk' = \mathsf{ACE.GenDK}(sk, C_2)$*.*
3. *The challenger computes* $ct_i = \mathsf{ACE.Enc}(ek, m_i^b)$ *for* $i \in [\ell(\lambda)]$*.*
4. *The challenger samples strings* r_i *for all* $i \in [\ell(\lambda)]$*, each the same length as the ciphertext length of* ACE *and each is sampled uniformly at random.*
5. *The challenger samples* $b \leftarrow \{0,1\}$*. If* $b = 0$*, it submits* $ct_1, \ldots, ct_{\ell(\lambda)}$ *to the adversary. If* $b = 1$*, it submits* $r_1, \ldots, r_{\ell(\lambda)}$ *to the adversary.*
6. *The adversary outputs a bit* b'*.*
7. *The challenger outputs* 1 *if* $b = b'$*, and outputs* 0 *otherwise.*

An asymmetrically constrained encapsulation scheme ACE *is said to satisfy* $\ell(\lambda)$-pseudorandom ciphertext security *if for any QPT adversary* $\mathcal{A}$*,*

$$\Pr\left[\mathsf{PRCiphertextGame}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Finally, we introduce our new security notion, *steganographic ciphertext security.*

Definition 11 (Steganographic Asymmetrically Constrained Encapsulation)**.** *Let* $\ell(\lambda)$*,* $c_{punc}(\lambda)$*,* $n(\lambda)$ *(as in Definition 8),* $c_{samp}(\lambda)$*, be parameters that are polynomials in* λ*. We define an admissible distribution* $\mathcal{D}$[12] *to be a distribution with a sampler circuit of size at most* $c_{samp}(\lambda)$ *and with min-entropy at least* $k(\lambda) \geq 4 \cdot n(\lambda)$*. Unless otherwise specified, we will implicitly mean an admissible distribution when we are considering a distribution for steganography.*

A ***steganographic asymmetrically constrained encapsulation scheme*** ACE *is an ACE scheme parametrized by* $c_{samp}(\lambda)$*,* $\ell(\lambda)$*,* $c_{punc}(\lambda)$*,* $n(\lambda)$*,* $k(\lambda)$ *with the following additional algorithms and guarantees.*

- $\mathsf{StegEnc}(ek, m, \langle\mathcal{D}\rangle)$*: Takes as input an encapsulation key* ek*, a message* m *and a sampler circuit* $\mathcal{D}$*, and outputs a ciphertext.*
- $\mathsf{StegDec}(dk, ct)$*: An* ***efficient*** *algorithm that takes as input a decapsulation key* dk *and a ciphertext* ct*, outputs a message* m *or* $\perp$*.*

We require the usual correctness requirements of an ACE scheme. Additionally, we require the following.

[12] We will overload the notation $\mathcal{D}$ to denote both the distribution itself and the sampling circuit for this distribution.

Correctness of Steganographic Encryption: *We require that for all (admissible) distributions $\mathcal{D}$ and all messages m, $\mathsf{ACE.StegEnc}(ek, m, \mathcal{D})$ runs in time $t_{steg}(\lambda)$ (which is a function that depends on $n(\lambda), |\mathsf{supp}(\mathsf{D})|$ and the desired security level) and satisfies the following.*

$$\Pr\left[\mathsf{ACE.StegDec}(dk, \mathsf{ACE.StegEnc}(ek, m, \langle\mathcal{D}\rangle)) = m : \begin{array}{c} sk \leftarrow \mathsf{ACE.Setup}(1^\lambda) \\ ek = \mathsf{ACE.GenEK}(sk, \mathsf{FALSE}) \\ dk = \mathsf{ACE.GenDK}(sk, \mathsf{FALSE}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Security: *We require that* ACE *satisfies puncture-hiding security and $\ell(\lambda)$-pseudorandom ciphertext security against any quantum adversary that runs in time $\mathsf{poly}(\lambda) \cdot t_{steg}(\lambda)$.*

Steganographic Ciphertext Security: *We require that* ACE *satisfies steganographic ciphertext security (defined later).*

Definition 12 (Steganographic Ciphertext Security)**.** *Consider the following game between a challenger and an adversary.*

$\underline{\mathsf{StegCiphertextGame}_{\mathcal{A}}(1^\lambda)}$

1. *The adversary $\mathcal{A}$ outputs two admissible circuits C_1, C_2 and $\ell(\lambda)$ messages $(m_1, \ldots, m_{\ell(\lambda)})$ such that $C_1(m_i) = C_2(m_i) = \mathsf{TRUE}$ for all $i \in [\ell(\lambda)]$.*
2. *The adversary outputs an (admissible) sampler $\mathcal{D}$.*
3. *The challenger samples $sk \leftarrow \mathsf{ACE.Setup}(1^\lambda)$ and computes $ek = \mathsf{ACE.GenEK}(sk, \mathsf{FALSE})$ $ek' = \mathsf{ACE.GenEK}(sk, C_1)$, $dk' = \mathsf{ACE.GenDK}(sk, C_2)$.*
4. *The challenger computes $ct_i = \mathsf{ACE.StegEnc}(ek, m_i, \langle\mathcal{D}\rangle)$ for $i \in [\ell(\lambda)]$.*
5. *The challenger samples strings $samp_i \leftarrow \mathcal{D}$ for all $i \in [\ell(\lambda)]$.*
6. *The challenger samples $b \leftarrow \{0,1\}$. If $b = 0$, it submits $ct_1, \ldots, ct_{\ell(\lambda)}$ to the adversary. If $b = 1$, it submits $samp_1, \ldots, samp_{\ell(\lambda)}$ to the adversary.*
7. *The adversary outputs a bit b'.*
8. *The challenger outputs 1 if $b = b'$, and outputs 0 otherwise.*

An asymmetrically constrained encapsulation scheme ACE *is said to satisfy* steganographic ciphertext security *if for any QPT adversary $\mathcal{A}$,*

$$\Pr\left[\mathsf{StegCiphertextGame}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Theorem 8. *Assuming the existence of subexponentially secure indistinguishability obfuscation and one-way functions, for any polynomials $n(\lambda), c_{punc}(\lambda), c_{samp}(\lambda), \ell(\lambda)$, there exits a subexponentially secure steganographic asymmetrically constrainable encryption scheme.*

Proof. We give our construction in Section 7.2. The proof of steganographic ciphertext security is given in Section 7.3. Steganographic correctness is immediate. All the other correctness and security notions follow from [ÇG25]. All the assumed primitives can be instantiated assuming existence of subexponentially secure indistinguishability obfuscation and one-way functions for any subexponential function, as explained in Section 7.2, Section 7.3 and [ÇG25]. □

7.2 Construction

In this section, we give our construction for a steganographic ACE scheme. Our construction is similar to the construction [CHJV15], except for two key differences: (i) addition of the steganographic encryption/decryption algorithms (which naturally also requires a novel security proof), (ii) choosing the underlying primitives and their parameters in a delicate way that is required for our new security notions.

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

1. Sample $K_1 \leftarrow \mathsf{PRF}_1.\mathsf{KeyGen}(1^\lambda)$ and $K_2 \leftarrow \mathsf{PRF}_2.\mathsf{KeyGen}(1^\lambda)$.
2. Sample $seed \leftarrow \{0,1\}^{\mathsf{se}(\lambda)}$.
3. Output $sk = (K_1, K_2, seed)$.

$\underline{\mathsf{ACE.GenEK}(sk, C)}$

1. Parse $(K_1, K_2, seed) = sk$.
2. Sample $\hat{P} \leftarrow i\mathcal{O}(P)$ where P is the following program.

   $\underline{P(m)}$
   Hardcoded: K_1, K_2, C
   1. Check if $C(m) = \mathsf{TRUE}$. If so, output $\perp$ and terminate.
   2. Compute $\alpha = \mathsf{PRF}_1.\mathsf{Eval}(K_1, m)$.
   3. Compute $\beta = \mathsf{PRF}_2.\mathsf{Eval}(K_2, \alpha) \oplus m$.
   4. Output $\alpha || \beta$.

3. Output $(\hat{P}, seed)$.

$\underline{\mathsf{ACE.GenDK}(sk, C)}$

1. Parse $(K_1, K_2, seed) = sk$.
2. Sample $\hat{P} \leftarrow i\mathcal{O}(P)$ where P is the following program.

   $\underline{P(ct)}$
   Hardcoded: K_1, K_2, C
   1. Check if $C(m) = \mathsf{TRUE}$. If so, output $\perp$ and terminate.
   2. Parse $\alpha || \beta = ct$.
   3. Compute $m = \mathsf{PRF}_2.\mathsf{Eval}(K_2, \alpha) \oplus \beta$.
   4. Check if $\alpha = \mathsf{PRF}_1.\mathsf{Eval}(K_1, m)$. If not, output $\perp$ and terminate.
   5. Output m.

3. Output $(\hat{P}, seed)$.

$\underline{\mathsf{ACE.Enc}(ek, m)}$

1. Parse $(\hat{P}, seed) = ek$.
2. Output $\hat{P}(m)$.

$\underline{\mathsf{ACE.Dec}(dk, ct)}$

1. Parse $(\hat{P}, seed) = dk$.
2. Output $\hat{P}(ct)$.

$\underline{\mathsf{ACE.StegEnc}(ek, m)}$

1. Parse $(\hat{P}, seed) = ek$.
2. Set $ict = \mathsf{ACE.Enc}(ek, m)$.
3. Set $cnt = 1$.
4. Repeat the following as long as $cnt \leq \ell$: Sample $s' \leftarrow \mathcal{D}$. If $\mathsf{Ext}(seed, s') = ict$, output s' and terminate, otherwise increase cnt by one and continue.
5. Output $\perp$ if not already terminated.

$\underline{\mathsf{ACE.StegDec}(dk, ct)}$

1. Parse $(\hat{P}, seed) = dk$.
2. Output $\hat{P}(\mathsf{Ext}(seed, ct))$.

7.3 Proof of Steganographic Ciphertext Security

We first start with two technical lemmata.

Lemma 12 (Infinite Reverse Resampling Lemma)**.** *Let $\mathcal{D}$ be any distribution, S be any set and $f : \mathsf{supp}(\mathcal{D}) \to S$ be any function. Consider the distribution $\mathcal{D}'$ defined as follows.*

$\underline{\mathcal{D}'}$

1. *Sample $s \leftarrow \mathcal{D}$.*
2. *Set $y = f(s)$.*
3. *Repeat the following until success: Sample $s' \leftarrow \mathcal{D}$. If $f(s') = y$, output s' and terminate, otherwise continue.*

Then, $\mathcal{D} \equiv \mathcal{D}'$.

Proof. Fix any $x \in \mathsf{supp}(\mathcal{D})$ and we will show that $\Pr_{\mathcal{D}'}[x] = \Pr_{\mathcal{D}}[x]$, which implies $\mathcal{D} \equiv \mathcal{D}'$.

We write

$$\begin{aligned}
\Pr_{\mathcal{D}'}[x] &= \sum_{s \in \mathsf{supp}(\mathcal{D})} \Pr_{\mathcal{D}}[s] \cdot \lim_{k \to \infty} \left(\sum_{i=0}^{k} (\Pr_{s_1 \leftarrow \mathcal{D}}[f(s_1) \neq f(s)])^i \Pr_{s_2 \leftarrow D}[s_2 = x \wedge f(x) = f(s)] \right) \\
&= \sum_{s \in \mathsf{supp}(\mathcal{D})} \Pr_{\mathcal{D}}[s] \cdot \Pr_{s_2 \leftarrow D}[s_2 = x \wedge f(x) = f(s)] \cdot \frac{1}{1 - \Pr_{s_1 \leftarrow \mathcal{D}}[f(s_1) \neq f(s)]} \\
&= \sum_{s \in \mathsf{supp}(\mathcal{D})} \Pr_{\mathcal{D}}[s] \cdot \Pr_{s_2 \leftarrow D}[s_2 = x \big| f(s_2) = f(s)] \\
&= \sum_{y \in f(\mathsf{supp}(\mathcal{D}))} \sum_{\substack{s \in \mathsf{supp}(\mathcal{D}): \\ f(s) = y}} \Pr_{\mathcal{D}}[s] \cdot \Pr_{s_2 \leftarrow D}[s_2 = x \big| f(s_2) = f(s)] \\
&= \sum_{y \in f(\mathsf{supp}(\mathcal{D}))} \sum_{\substack{s \in \mathsf{supp}(\mathcal{D}): \\ f(s) = y}} \Pr_{\mathcal{D}}[s] \cdot \Pr_{s_2 \leftarrow D}[s_2 = x \big| f(s_2) = y] \\
&= \sum_{y \in f(\mathsf{supp}(\mathcal{D}))} \Pr_{s_2 \leftarrow D}[s_2 = x \big| f(s_2) = y] \cdot \left(\sum_{\substack{s \in \mathsf{supp}(\mathcal{D}): \\ f(s) = y}} \Pr_{\mathcal{D}}[s] \right) \\
&= \sum_{y \in f(\mathsf{supp}(\mathcal{D}))} \Pr_{s_2 \leftarrow D}[s_2 = x \big| f(s_2) = y] \cdot \Pr_{s \leftarrow \mathcal{D}}[f(s) = y] = \Pr_{s_2 \leftarrow \mathcal{D}}[s_2 = x]
\end{aligned}$$

The first line is by the Law of Total probability and the second line is by the power series solution and the other lines should be self-evident. □

Lemma 13 (Truncated Reverse Resampling Lemma)**.** *Let $\mathcal{D}$ be any distribution, S be any set and $f : \mathsf{supp}(\mathcal{D}) \to S$ be any function. Let $\epsilon > 0$ and $t_{limit} = \lceil \frac{2\left(\log(4)n + \log\left(\frac{1}{\epsilon}\right)\right) \cdot |\mathsf{supp}(\mathcal{D})|)}{\epsilon} \rceil$. Consider the distribution $\mathcal{D}'$ defined as follows.*

$\underline{\mathcal{D}'}$

1. *Sample $s \leftarrow \mathcal{D}$.*
2. *Set $y = f(s)$.*
3. *Set $cnt = 1$.*
4. *Repeat the following as long as $cnt \leq t_{limit}$: Sample $s' \leftarrow \mathcal{D}$. If $f(s') = y$, output s' and terminate, otherwise increase cnt by one and continue.*
5. *Output $\perp$ if not already terminated.*

Then, $\mathcal{D} \approx_\epsilon \mathcal{D}'$ and $\Pr_{\mathcal{D}'}[f(s) = y] \geq 1 - \epsilon$.

Proof. Define E to be the event that during execution of the sampler $\mathcal{D}'$, the preimage search

succeeds, that is, the counter does not reach $t_{limit}+1$. We write

$$\begin{aligned}
|\mathcal{D}-\mathcal{D}'|_1 &= \sum_{x\in\mathsf{supp}(\mathcal{D})} |\Pr_{D}[x]-\Pr_{D'}[x]| \\
&= \sum_{x\in\mathsf{supp}(\mathcal{D})} |\Pr_{D}[x]-\Pr_{x'\leftarrow D'}[x'=x|E]\cdot\Pr_{\mathcal{D}'}[E]-\Pr_{x'\leftarrow D'}[x'=x|\overline{E}]\cdot\Pr_{\mathcal{D}'}[\overline{E}]| \\
&= \sum_{x\in\mathsf{supp}(\mathcal{D})} |\Pr_{D}[x]-\Pr_{x'\leftarrow D'}[x=x'|E]\cdot\Pr_{\mathcal{D}'}[E]| \\
&= \sum_{x\in\mathsf{supp}(\mathcal{D})} |\Pr_{D}[x]-\Pr_{x'\leftarrow D}[x=x']\cdot\Pr_{\mathcal{D}'}[E]| \\
&= (1-\Pr_{\mathcal{D}'}[E])\cdot\sum_{x\in\mathsf{supp}(\mathcal{D})}\Pr_{D}[x]
\end{aligned}$$

The third line is due to the fact that when $\overline{E}$ occurs, output is $\perp$. The fourth line is by Lemma 12.

Set $th=\frac{\epsilon}{2\cdot|\mathsf{supp}(\mathcal{D})|}$. Now we have

$$\begin{aligned}
\Pr_{\mathcal{D}'}[\overline{E}] &\le \sum_{x\in\mathsf{supp}(\mathcal{D})}\Pr_{\mathcal{D}}[x]\cdot(1-\Pr_{\mathcal{D}}[x])^{t_{limit}} \\
&= \sum_{\substack{x\in\mathsf{supp}(\mathcal{D}):\\ \Pr_{\mathcal{D}}[x]\le th}}\Pr_{\mathcal{D}}[x]\cdot(1-\Pr_{\mathcal{D}}[x])^{t_{limit}} + \sum_{\substack{x\in\mathsf{supp}(\mathcal{D}):\\ \Pr_{\mathcal{D}}[x]> th}}\Pr_{\mathcal{D}}[x]\cdot(1-\Pr_{\mathcal{D}}[x])^{t_{limit}} \\
&\le |\mathsf{supp}(\mathcal{D})|\cdot(th+e^{-t_{limit}\cdot th}) \le \epsilon
\end{aligned}$$

□

Now we prove security through a sequence of hybrids, each of which is obtained by modifying the previous one. For simplicity we will only prove the case $\ell=1$, but the general case follows similarly (given that obfuscated program sizes will depend on ℓ). We first define our hybrids and then we will show their indistinguishability.

Hyb_0**:** The original game $\mathsf{StegCiphertextGame}_{\mathcal{A}}(1^\lambda)$.

Hyb_1**:** We modify the way the challenger computes the punctured key ek' that will be submitted to the adversary: Instead of setting $ek=\mathsf{ACE.GenEK}(sk,C_1)$, the challenger now sets $ek'=(\hat{P}_{enc},seed)$ where

- $K_1'\leftarrow\mathsf{PRF}_1.\mathsf{Punc}(K_1,m^*)$.
- $\alpha^*=\mathsf{PRF}_1.\mathsf{Eval}(K_1,m^*)$.
- $\beta^*=\mathsf{PRF}_1.\mathsf{Eval}(K_2,\alpha^*)$. (Note that $\mathsf{ACE.Enc}(ek,m)$ is $\alpha^*||\beta^*$ during execution of $\mathsf{StegEnc}$)
- $\hat{P}\leftarrow i\mathcal{O}(P'_{enc})$ where P'_{enc} is the following program.

$P'_{enc}(m)$

Hardcoded: K_1', K_2, C_1

1. Check if $C_1(m)=\mathsf{TRUE}$. If so, output $\perp$ and terminate.
2. Compute $\alpha=\mathsf{PRF}_1.\mathsf{Eval}(K_1',m)$.

3. Compute $\beta = \mathsf{PRF}_2.\mathsf{Eval}(K_2, \alpha) \oplus m$.
4. Output $\alpha||\beta$.

$\underline{\mathsf{Hyb}_2}$: We modify the way the challenger computes the punctured key dk' that will be submitted to the adversary: Instead of setting $dk = \mathsf{ACE.GenDK}(sk, C_2)$, the challenger now sets $dk' = (\hat{P}_{dec}, seed)$ where $\hat{P} \leftarrow i\mathcal{O}(P'_{dec})$ where P'_{dec} is the following program.

$\underline{P'_{dec}(ct)}$

Hardcoded: K'_1, K_2, C

1. Check if $C(m) = \mathsf{TRUE}$. If so, output $\perp$ and terminate.
2. Parse $\alpha||\beta = ct$.
3. Compute $m = \mathsf{PRF}_2.\mathsf{Eval}(K_2, \alpha) \oplus \beta$.
4. Check if $\alpha = \mathsf{PRF}_1.\mathsf{Eval}(K'_1, m)$. If not, output $\perp$ and terminate.
5. Output m.

$\underline{\mathsf{Hyb}_3}$: We now sample $\alpha^* \leftarrow \{0,1\}^{3n(\lambda)} \setminus \mathsf{Img}(\mathsf{PRF}_1.\mathsf{Eval}(K_1, \cdot))$.

$\underline{\mathsf{Hyb}_4}$: When sampling ek', we now sample $K'_2 \leftarrow \mathsf{PRF}_2.\mathsf{Punc}(K_2, \{\alpha^*, r_1^*\})$ where $r_1^* \leftarrow \{0,1\}^{3n(\lambda)} \setminus \mathsf{Img}(\mathsf{PRF}_1.\mathsf{Eval}(K_1, \cdot))$ and use K'_2 instead of K_2.

$\underline{\mathsf{Hyb}_5}$: We modify the way the challenger computes the punctured key dk' that will be submitted to the adversary: Instead of setting $dk = \mathsf{ACE.GenDK}(sk, C_2)$, the challenger now sets $dk' = (\hat{P}_{dec}, seed)$ where $\hat{P} \leftarrow i\mathcal{O}(P''_{dec})$ where P''_{dec} is the following program.

$\underline{P''_{dec}(ct)}$

Hardcoded: $K'_1, K_2, C, \alpha^*, r_1^*$

1. Check if $C(m) = \mathsf{TRUE}$. If so, output $\perp$ and terminate.
2. Check if $\alpha = \alpha^*$ or r_1^*. If so, output $\perp$ and terminate.
3. Parse $\alpha||\beta = ct$.
4. Compute $m = \mathsf{PRF}_2.\mathsf{Eval}(K_2, \alpha) \oplus \beta$.
5. Check if $\alpha = \mathsf{PRF}_1.\mathsf{Eval}(K'_1, m)$. If not, output $\perp$ and terminate.
6. Output m.

$\underline{\mathsf{Hyb}_6}$: When sampling dk', we now sample $K'_2 \leftarrow \mathsf{PRF}_2.\mathsf{Punc}(K_2, \{\alpha^*, r_1^*\})$ where $r_1^* \leftarrow \{0,1\}^{3n(\lambda)}$ and use K'_2 instead of K_2.

$\underline{\mathsf{Hyb}_7}$: We now sample $\beta^* \leftarrow \{0,1\}^{n(\lambda)}$.

$\underline{\mathsf{Hyb}_8}$**:** We now sample $r_1^* \leftarrow \{0,1\}^{3n(\lambda)}$.

Lemma 14. $\mathsf{Hyb}_0 \approx \mathsf{Hyb}_1$.

Proof. The circuits in the two hybrids are equivalent by punctured key correctness of the PRF scheme. The result follows by $i\mathcal{O}$ security. □

Lemma 15. $\mathsf{Hyb}_1 \approx \mathsf{Hyb}_2$.

Proof. The circuits in the two hybrids are equivalent by punctured key correctness of the PRF scheme. The result follows by $i\mathcal{O}$ security. □

Lemma 16. $\mathsf{Hyb}_2 \approx_{2^{-2n(\lambda)}} \mathsf{Hyb}_3$.

Proof. Follows by punctured key security of the PRF scheme and the fact that the image set of $\mathsf{PRF}_1.\mathsf{Eval}_1(K_1, \cdot)$ has size $2^{n(\lambda)}$. □

Lemma 17. $\mathsf{Hyb}_3 \approx \mathsf{Hyb}_4$.

Proof. Follows by PRF punctured key correctness and $i\mathcal{O}$ security. □

Lemma 18. $\mathsf{Hyb}_4 \approx \mathsf{Hyb}_5$.

Proof. Follows by $i\mathcal{O}$ security. □

Lemma 19. $\mathsf{Hyb}_5 \approx \mathsf{Hyb}_6$.

Proof. Follows by PRF punctured key correctness and $i\mathcal{O}$ security. □

Lemma 20. $\mathsf{Hyb}_6 \approx \mathsf{Hyb}_7$.

Proof. Follows by PRF punctured key security. □

Lemma 21. $\mathsf{Hyb}_7 \approx_{2^{-2n(\lambda)}} \mathsf{Hyb}_8$.

Proof. Immediate by an elementary probability calculation. □

The above argument shows that in the experiment $\mathsf{StegCiphertextGame}_{\mathcal{A}}(1^\lambda)$, during the computation of $\mathsf{ACE.StegEnc}$, we can replace the value $ict = \mathsf{ACE.Enc}(ek, m)$ with a truly random value r. Then, we do the following argument. We replace r with $\mathsf{Ext}(seed, s^*)$ where $s^* \leftarrow \mathcal{D}$, and these hybrids are indistinguishable by extractor security. Finally, we can replace the outcome of $\mathsf{ACE.StegEnc}$ with a true sample from s'' from $\mathcal{D}$ by Lemma 13.

8 Acknowledgments

Part of this work done was done while AÇ was an intern at NTT Research and part of this work was done while AÇ was being supported by the following grants of VG: NSF award 1916939, DARPA SIEVE program, a gift from Ripple, a DoE NETL award, a JP Morgan Faculty Fellowship, a PNC center for financial services innovation award, and a Cylab seed funding award.

References

[Aar09] Scott Aaronson. Quantum copy-protection and quantum money. In *2009 24th Annual IEEE Conference on Computational Complexity*, pages 229–242, 2009.

[Aar16] Scott Aaronson. The complexity of quantum states and transformations: From quantum money to black holes, 2016.

[AB24] Prabhanjan Ananth and Amit Behera. A modular approach to unclonable cryptography. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part VII*, volume 14926 of *Lecture Notes in Computer Science*, pages 3–37, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[AL21] Prabhanjan Ananth and Rolando L. La Placa. Secure software leasing. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part II*, volume 12697 of *Lecture Notes in Computer Science*, pages 501–530, Zagreb, Croatia, October 17–21, 2021. Springer, Cham, Switzerland.

[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, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 526–555, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland.

[ÇG24a] Alper Çakan and Vipul Goyal. Unclonable cryptography with unbounded collusions and impossibility of hyperefficient shadow tomography. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024: 22nd Theory of Cryptography Conference, Part III*, volume 15366 of *Lecture Notes in Computer Science*, pages 225–256, Milan, Italy, December 2–6, 2024. Springer, Cham, Switzerland.

[CG24b] Andrea Coladangelo and Sam Gunn. How to use quantum indistinguishability obfuscation. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *56th Annual ACM Symposium on Theory of Computing*, pages 1003–1008, Vancouver, BC, Canada, June 24–28, 2024. ACM Press.

[ÇG25] Alper Çakan and Vipul Goyal. How to copy-protect all puncturable functionalities without conjectures: A unified solution to quantum protection. *Cryptology ePrint Archive*, 2025.

[ÇGLZR24] Alper Çakan, Vipul Goyal, Chen-Da Liu-Zhang, and João Ribeiro. Unbounded leakage-resilience and intrusion-detection in a quantum world. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024: 22nd Theory of Cryptography Conference, Part II*, volume 15365 of *Lecture Notes in Computer Science*, pages 159–191, Milan, Italy, December 2–6, 2024. Springer, Cham, Switzerland.

[CHJV15] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Succinct garbling and indistinguishability obfuscation for RAM programs. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 429–437, Portland, OR, USA, June 14–17, 2015. ACM Press.

[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, Part I, volume 12825 of *Lecture Notes in Computer Science*, pages 556–584, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland.

[DPVR12] Anindya De, Christopher Portmann, Thomas Vidick, and Renato Renner. Trevisan's extractor in the presence of quantum side information. *SIAM Journal on Computing*, 41(4):915–940, 2012.

[NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 475–484, New York, NY, USA, May 31 – June 3, 2014. ACM Press.

[Wie83] Stephen Wiesner. Conjugate coding. *SIGACT News*, 15(1):78–88, jan 1983.

[Zha12] Mark Zhandry. How to construct quantum random functions. In *53rd Annual Symposium on Foundations of Computer Science*, pages 679–687, New Brunswick, NJ, USA, October 20–23, 2012. IEEE Computer Society Press.

[Zha20] Mark Zhandry. Schrödinger's pirate: How to trace a quantum decoder. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part III*, volume 12552 of *Lecture Notes in Computer Science*, pages 61–91, Durham, NC, USA, November 16–19, 2020. Springer, Cham, Switzerland.