

The Power of Permutable PRPs:

Towards Trapdoor Permutations with Full Key and Message Domains

Shany Ben-David
shany.ben-david@biu.ac.il
Bar-Ilan University

Eylon Yogev
eylon.yogev@biu.ac.il
Bar-Ilan University

October 4, 2026

Abstract

Shmueli and Zhandry introduced *permutable PRPs* and used them to construct trapdoor permutations with a full message domain, but a sparse public-key space. We demonstrate the further power of permutable PRPs by constructing puncturable trapdoor permutations with full message and key domains in the common-reference-string (CRS) model. Our construction assumes a secure permutable PRP for general swaps, indistinguishability obfuscation, and an injective length-doubling pseudorandom generator, all secure against polynomial-time adversaries. This takes a step towards trapdoor permutations with full key and message domains by allowing a shared setup.

A one-time setup produces a CRS and a master trapdoor. For every honestly generated CRS, every ℓ -bit string indexes a permutation of $\{0, 1\}^n$, and anyone can sample a public key uniformly without validation or rejection sampling. The master trapdoor derives an inversion trapdoor for every key. The same family supports two levels of puncturing: a master trapdoor punctured at k^* derives trapdoors for every other key, while an individual trapdoor punctured at y^* inverts every other output under k^* . Even given both punctured trapdoors, inverting y^* under k^* remains hard for uniformly sampled k^* and y^* . To our knowledge, this is the first trapdoor permutation family in the CRS model with full message and key domains and a master trapdoor that derives an inversion trapdoor for every key.

We give three applications. In the random-oracle model, the family yields deterministic identity-based encryption with zero ciphertext expansion and adaptive weak-source pseudorandomness for message sources satisfying our computational hardness condition. A punctured secret key decrypts every ciphertext except the challenge under the selected identity, providing a chosen-ciphertext-style guarantee. The family also yields identity-based unique signatures with adaptive unforgeability in the random-oracle model. In addition, our puncturable trapdoors give a direct construction of hard-on-average END-OF-LINE instances, establishing PPAD hardness without passing through SINK-OF-VERIFIABLE-LINE.

Keywords: trapdoor permutations, puncturable permutations, full key domains, indistinguishability obfuscation

Contents

1	Introduction	1
1.1	Our Results	2
1.2	Applications	3
1.3	Deterministic length-preserving identity-based encryption	4
1.4	Direct PPAD hardness	4
1.5	Related Work	5
1.6	Independent and concurrent work	7
1.7	Paper organization	7
2	Technical overview	8
2.1	Permutable PRPs	8
2.2	Puncturable PRPs - definitions	9
2.3	TDP with puncturable trapdoor	10
2.4	TDPs with full key and message domain	11
2.5	Permutable PRPs with a full key domain	14
2.6	Puncturable PRPs - security proof	16
2.7	Deterministic length-preserving IBE	18
2.8	Direct hardness of End-of-Line	18
3	Preliminaries	20
3.1	Indistinguishability Obfuscation	20
3.2	Pseudorandom Generators	20
3.3	Permutable PRP	21
4	How to puncture PRPs	24
4.1	Definition	24
4.2	Construction and proof	25
5	TDP with puncturable-trapdoor	35
5.1	Definition	36
5.2	Construction	38
5.3	Proof of Theorem 5.4	38
5.4	Proof of Theorem 5.5	42
5.5	Proof of Corollary 5.6	45
5.6	Application - PPAD Hardness	46
6	TDP with a full key domain in the CRS model	50
6.1	Definition	50
6.2	Construction	52
6.3	Proof of Theorem 6.5	54
6.4	Proof of Theorem 6.6	60
6.5	Proof of Corollary 6.7	61

7	Deterministic identity-based encryption	64
7.1	Definition	64
7.2	Construction and proof	65
8	Identity-based unique signatures	72
8.1	Definition	72
8.2	Construction and proof	73
9	Permutable PRP with a full key domain in the CRS model	77
9.1	Definition	77
9.2	Construction and proof	80
	References	96
A	Generalization of the sparse trigger lemma	98
	Acknowledgments	102

1 Introduction

Permutations combine two properties central to cryptography: every output has a unique preimage, yet finding that preimage may be computationally hard. A *one-way permutation* is publicly evaluable but hard to invert on a random image. A *trapdoor permutation* (TDP) additionally admits efficient inversion given a secret trapdoor. A *pseudorandom permutation* (PRP), in contrast, uses a secret evaluation key and is computationally indistinguishable from a uniformly random permutation under oracle access. These notions support different uses of the same bijective structure, from public-key cryptography to symmetric encryption.

The idea of a trapdoor function dates to the birth of public-key cryptography [DH76] and was formalized by Yao [Yao82]. Trapdoor permutations give generic constructions of public-key encryption [GM84] and signatures, and suitable variants underpin non-interactive zero-knowledge proofs in the common-reference-string (CRS) model [BFM88; FLS99]. One-way permutations also yield non-interactive commitments. By comparison, the classical construction of commitments from one-way functions uses interaction [Nao91]. For these applications, the existence of an efficiently evaluable bijection is only part of the requirement: how its domain and public key are represented and sampled can be equally important.

Full-domain permutations. In a general formulation, key generation outputs a public index pk and a trapdoor td , and the index determines a domain D_{pk} on which public evaluation is a bijection. The set D_{pk} need not be all bit strings of a given length: it may be sparse, hard to recognize, or accessible only through a randomized sampler. This led to a hierarchy of refinements, including *enhanced* and *doubly enhanced* TDPs [GR13], as well as recognizability or certifiability requirements needed in some NIZK transformations [CL18].

When the domain is $\{0,1\}^n$, membership is trivial and uniform sampling simply returns the sampler’s random coins. The corresponding sampling-related enhancements then follow automatically. A separate issue concerns the public keys: even if every permutation acts on $\{0,1\}^n$, valid public descriptions may occupy only a sparse subset of all strings of the prescribed length. We therefore distinguish two requirements:

- *Full message domain.* Every valid key defines a permutation of all n -bit strings.
- *Full key domain.* Every string of the prescribed key length is a valid key.

The first requirement simplifies sampling messages, and the second makes it possible to sample a valid public key by choosing a uniform bit string. Neither requirement alone implies the other.

Obfuscating permutations. A natural approach to constructing a trapdoor permutation is to publish an obfuscated forward evaluator for a PRP and retain its key for inversion. The difficulty is proving that the published program hides the inverse. General-purpose virtual-black-box obfuscation is impossible [BGIRSVY01], while indistinguishability obfuscation (iO) only guarantees that equivalent circuits have indistinguishable obfuscations. It does not directly guarantee that obfuscating an arbitrary PRP yields a one-way permutation.

Many iO constructions use puncturable PRFs to remove a secret value from an obfuscated program while preserving evaluation everywhere else. Extending this approach to permutations requires care: their outputs are constrained by bijectivity, and forward and inverse evaluation should typically remain consistent. In particular, after all but one image are revealed, the remaining image is determined.

Prior constructions nevertheless obtained trapdoor permutations from iO and one-way functions [BPW16; GPSZ17]. Their constructions leave restrictions on the message domain and its sampling interface. Moreover, Asharov and Segev showed that a broad class of black-box constructions from iO and one-way functions cannot simultaneously achieve domain invariance for both messages and keys [AS16].

Permutable PRPs. A recent work of Shmueli and Zhandry [SZ25] introduced *permutable PRPs*, an analogue of puncturability tailored to permutations. Informally, given a secret PRP key k and a public permutation Γ from a supported class, one can derive a key for either Π_k or $\Gamma \circ \Pi_k$, together with the corresponding inverse. The two derived-key distributions are computationally indistinguishable. Input permutability instead allows composition on the input side, and bilateral permutability allows both. We use these interfaces for transpositions, which swap two points and fix every other point. They allow a proof to replace an explicit swap in a program by a permuted key implementing the same functionality, and then use permutability security to remove the swap computationally. Shmueli and Zhandry constructed permutable PRPs from (subexponentially secure) iO and one-way functions and showed that obfuscating them yields a trapdoor one-way permutation with full message domain. Their public keys, being obfuscated programs, still occupy a sparse subset of their ambient bit-string space.

Full key domain. A full message domain makes every message string usable, but applications may also need to use externally supplied strings as public keys. For example, when a random oracle maps identities to keys, a sparse key domain gives no guarantee that its outputs describe valid permutations. A full key domain removes this obstacle: every output of the prescribed length is a valid key, with no additional encoding or rejection sampling. Combined with a master trapdoor that derives an inversion trapdoor for each key, this allows identities to index permutations directly and gives a natural route to identity-based encryption. The two domain requirements thus serve complementary purposes: a full key domain accommodates arbitrary key strings, while a full message domain ensures that the resulting permutations preserve the entire message space.

1.1 Our Results

We show that the power of permutable PRPs extends beyond the full-message-domain trapdoor permutations of Shmueli and Zhandry. Our main result is a single family of trapdoor permutations in the CRS model that combines full message and key domains with puncturing at both the master-trapdoor and individual-trapdoor levels.

TDPs with full key and message domains. We address the remaining key-domain sparsity by allowing a shared setup. A one-time setup produces a public CRS and a secret master trapdoor. Anyone can then sample a per-instance key uniformly from a full bit-string space. We require security even when the adversary receives punctured versions of both the master trapdoor and an individual trapdoor, as specified below. The full-key-domain requirement applies to these per-instance keys under an honestly generated CRS. We do not require every ambient string to be a valid CRS. This qualification is essential: the structured public description resides in the setup, so the result is a step towards full key and message domains rather than a construction with both properties in the plain model.

Let N be the message-domain size and $\ell = \ell(\lambda, \log N)$ the key length. The basic interface is:

- (i) *Setup*. A one-time setup samples $(\text{crs}, \text{mk}) \leftarrow \text{Gen}(1^\lambda, N)$.

(ii) *Public evaluation.* Any string $k \in \{0, 1\}^\ell$ indexes the permutation

$$F_{\text{crs},k}(x) := \Pi(\text{crs}, k, x) \quad \text{over } [N].$$

(iii) *Trapdoor inversion.* The master trapdoor derives $\text{td}_k \leftarrow \text{Trapdoor}(\text{mk}, k)$, and

$$\Pi^{-1}(\text{td}_k, F_{\text{crs},k}(x)) = x.$$

For every honestly generated crs and *every* string k of the prescribed length, $F_{\text{crs},k}$ is a permutation of the full message domain $[N]$. Taking $N = 2^n$ gives the full bit-string message domain. Public evaluation requires only (crs, k) , and sampling k requires neither the master trapdoor nor a key-validation step. Correctness holds for every key.

We give a strong security notion which we call *two-level punctured security*. We augment the interface with two puncturing operations that restrict inversion capabilities at a challenge key k^* and output y^* :

- *Key-punctured master trapdoor.* $\text{mk}\{k^*\} \leftarrow \text{PuncMK}(\text{mk}, k^*)$ derives td_k for every $k \neq k^*$.
- *Output-punctured trapdoor.* $\text{td}_{k^*}\{y^*\} \leftarrow \text{PuncTD}(\text{td}_{k^*}, y^*)$ correctly inverts F_{crs,k^*} at every $y \neq y^*$.
- *Two-level punctured security.* Given the CRS, uniformly sampled k^* and y^* , and both restricted trapdoors, every efficient adversary has only negligible probability of computing $F_{\text{crs},k^*}^{-1}(y^*)$.

Ordinary one-wayness follows by withholding these additional capabilities. This two-level guarantee is essential: it protects the missing preimage even when the adversary combines the two restricted inversion capabilities. To the best of our knowledge, we give the first construction of trapdoor permutations in the CRS model that combines full message and per-instance key domains with two-level punctured security.

Theorem 1 (Informal). *Assuming a secure PPRP for general swaps, secure indistinguishability obfuscation, and a secure injective length-doubling PRG, there exist trapdoor one-way permutations in the CRS model with full message and per-instance key domains that satisfy the two-level punctured security described above.*

Our construction combines a method for encoding structured keys into a full key space with a TPD with a puncturable trapdoor obtained from permutable PRPs. TPD with a puncturable trapdoor also exists as a standalone primitive and protects the inverse of any output fixed independently of key generation. The full interface and security statements appear in Definition 6.1 and theorems 5.4 and 6.5; the construction and proof ideas are developed in Section 2. These assumptions require only security against polynomial-time adversaries. Sub-exponential security is needed only when instantiating the assumed general-swap PPRP from iO and one-way functions via Theorem 3.8.

In addition to our full key domain TDPs, we also construct full key domain permutable PRPs, see Section 9 for further details.

1.2 Applications

The full key domain and master trapdoor give identity-based encryption and unique signatures, while trapdoor puncturing alone gives a direct route to PPAD hardness.

1.3 Deterministic length-preserving identity-based encryption

In the random-oracle model, we obtain identity-based encryption (IBE) with deterministic encryption and zero ciphertext expansion. A random oracle H maps each identity id to a key $k = H(\text{id})$, and encryption is simply

$$\text{Enc}(\text{id}, m) = F_{\text{crs}, H(\text{id})}(m).$$

The full key domain guarantees that every oracle output is a valid key, so every identity indexes a permutation. The authority uses the master trapdoor to derive $\text{td}_{H(\text{id})}$ as the user’s decryption key. The full message domain ensures that the message and ciphertext spaces are both $[N]$. Thus, the *complete ciphertext* is exactly as long as the plaintext.

Exact length preservation and perfect correctness force encryption under each fixed identity to be deterministic. Indeed, the ciphertext supports of distinct messages must be nonempty and disjoint; when the message and ciphertext spaces have the same cardinality, each support must be a singleton. Ordinary indistinguishability for arbitrary messages is therefore impossible: an adversary can encrypt candidate messages and compare the results.

We instead prove *adaptive weak-source pseudorandomness*. For an admissible message source fixed independently of setup, an encryption under an adaptively chosen identity is indistinguishable from a uniform ciphertext. The adversary may extract secret keys for all other identities and receives the challenge identity’s secret key punctured at the displayed ciphertext in either experiment. This key decrypts every other ciphertext under that identity, giving a chosen-ciphertext-style interface. The source requirement is the computational hardness condition in Definition 3.4. A min-entropy bound alone does not suffice for our theorem. This security notion is distinct from standard PRIV1-ID-CCA security.

Theorem 2 (Informal). *Assuming a secure PPRP for general swaps, secure indistinguishability obfuscation, and a secure injective length-doubling PRG, there exists, in the random-oracle model, a perfectly correct deterministic IBE scheme with identical message and ciphertext spaces. The scheme satisfies adaptive weak-source pseudorandomness with punctured decryption keys for message sources satisfying our hardness condition.*

The formal statement appears in Theorem 7.2.

To the best of our knowledge, this is the first construction of deterministic IBE with zero ciphertext expansion. The construction illustrates the roles of all three properties of our family: the full key domain accommodates random-oracle outputs, the full message domain gives exact length preservation, and puncturing supplies the restricted decryption capability.

Identity-based unique signatures. Identity-based signatures allow verification using the signer’s identity and shared public parameters, with a trusted authority issuing identity-specific signing keys. Uniqueness means that each identity–message pair admits exactly one valid signature. We obtain such a scheme in the random-oracle model with perfect correctness and uniqueness under honestly generated public parameters, and adaptive unforgeability even against adversaries that request signing keys for other identities and signatures on chosen messages. See Section 8.

1.4 Direct PPAD hardness

The standalone TDP with puncturable trapdoor already suffices for this application. Recall that END-OF-LINE (EOL) asks for another endpoint in a directed graph described by successor and

predecessor circuits, given one designated starting point. Let F be a public permutation and fix a vertex y_0 independently of key generation. Use F as the successor circuit and its inverse punctured at y_0 as the predecessor circuit, returning y_0 at the puncture. This cuts the permutation cycle containing y_0 into a line whose other endpoint is $F^{-1}(y_0)$, while the other cycles defined by the permutation have no endpoints. Every solution therefore reveals the missing preimage. The precise EOL formulation and reduction appears in Construction 5.9 and theorem 5.10.

Hardness of PPAD from one-way functions and iO was already known from the work of Bitansky, Paneth, and Rosen [BPR15]. Their route first constructs a hard instance of SINK-OF-VERIFIABLE-LINE (SVL) and then uses a reduction based on reversible computation from SVL to a PPAD-complete line problem. Our reduction bypasses SVL entirely: the permutation and its punctured inverse are themselves the successor and predecessor circuits of the EOL instance. To our knowledge, this is the first direct construction of hard EOL instances from one-way functions and iO, with sub-exponential security used to instantiate the assumed PPRP.

1.5 Related Work

Trapdoor permutations and their domains. Classical number-theoretic constructions use algebraically structured domains and public keys. The general theory consequently distinguishes evaluation and inversion from domain sampling and recognition. Enhanced and doubly enhanced trapdoor permutations strengthen the guarantees associated with sampling [GR13], while certifiability addresses the validity of public descriptions. Canetti and Lichtenberg explain why these distinctions matter for applications to non-interactive zero knowledge [CL18].

The word *dense* has been used for two distinct properties in this literature. Haitner studied trapdoor permutations whose *message domains* are polynomially dense in an ambient bit-string space [Hai04]. By contrast, work beginning with De Santis and Persiano considered public-key systems with dense public descriptions [SP92]. This latter property was used, for example, to obtain uniformly distributed setup strings in universally composable protocols [CLOS02]. Depending on the formulation, density asks that honestly generated public descriptions be pseudorandom, or that a noticeable fraction of ambient strings be secure descriptions. These distributional requirements neither guarantee a valid permutation for every ambient key string nor provide a master trapdoor for all keys under a shared setup.

A still stronger assumption considered in the literature is that of enhanced, or doubly enhanced, trapdoor permutations with pseudorandom public keys [HMQU18]. Here the distribution of honestly generated public keys is computationally indistinguishable from uniform. The support of key generation may nevertheless be a proper subset of the ambient space, and inversion trapdoors are generated separately with the public keys. In particular, this assumption neither supplies a master trapdoor from which all per-key trapdoors can be derived nor establishes correctness for every ambient key string. Hofheinz, Müller-Quade, and Unruh also observed that no concrete candidates for this assumption were known. Morimae and Yamakawa later showed that classically secure full-domain trapdoor permutations suffice for non-verifiable remote state preparation and proofs of quantumness [MY23].

Identity-based and branch-indexed trapdoor functions. The closest structural precedent for our CRS abstraction comes from identity-based cryptography. Lattice trapdoors already allow a public setup and a master trapdoor to associate secret information with arbitrary public vectors or identities [GPV08]. Bellare, Kiltz, Peikert, and Waters formalized identity-based trapdoor functions,

in which public parameters define a deterministic injective map for every identity and a master secret derives an identity-specific inversion key [BKPW12]. This gives a full identity space and a master derivation mechanism. Their maps, however, are injective trapdoor functions whose ciphertext spaces are generally larger than their domains, not permutations of a common full message space.

Lossy trapdoor functions provide another related branch structure. In the all-but-one variant of Peikert and Waters, most branches are injective and trapdoor invertible, while a designated branch is lossy [PW08]. All-but-many trapdoor functions extend this idea to many lossy tags [Hof12]. These primitives deliberately place non-injective functions on some branches and generally use unequal domain and range spaces. In our construction every key indexes a permutation of the same full message space. Puncturing removes an inversion capability at one key or one output without changing the fact that the publicly evaluated map is a permutation.

Trapdoor permutations from obfuscation. Bitansky, Paneth, and Wichs, and independently Garg, Pandey, Srinivasan, and Zhandry, constructed trapdoor permutations from indistinguishability obfuscation and additional primitives [BPW16; GPSZ17]. These constructions do not provide a full message domain. On the negative side, Asharov and Segev proved a black-box barrier to obtaining one-way permutations with both full message and full key domains from one-way functions and iO [AS16].

Permutable PRPs were introduced by Shmueli and Zhandry [SZ25]. They used this primitive to construct a trapdoor one-way permutation over the full message domain, but a public key is still an obfuscated circuit and therefore belongs to a sparse set of valid descriptions. Our construction moves this remaining structured description into a reusable CRS and exposes a full domain of per-instance keys. The same construction supports puncturing both the master trapdoor at a key and each derived trapdoor at an output, with security even when both punctured objects are revealed together.

PPAD and CLS hardness. Bitansky, Paneth, and Rosen established PPAD hardness from one-way functions and iO by first constructing hard instances of SINK-OF-VERIFIABLE-LINE and then reducing them to a PPAD-complete line problem [BPR15]. Hubáček and Yogev subsequently showed that the SVL route yields the stronger conclusion of CLS hardness: they reduce SVL to END-OF-METERED-LINE, a problem contained in CLS [HY20].

Our standalone output-puncturing interface gives the successor and predecessor circuits of an END-OF-LINE instance directly. This direct construction does not supply the metering structure used in END-OF-METERED-LINE, so it does not establish CLS hardness. The contribution is the direct realization of the EOL circuits from a permutation and its punctured inverse; the stronger CLS conclusion remains a feature of the earlier SVL route.

Deterministic encryption. Deterministic public-key encryption and its appropriate high-entropy security notions were developed by Bellare, Boldyreva, and O’Neill [BBO07]. Their RSA-based construction also gives a length-preserving public-key cipher. Bellare, Kiltz, Peikert, and Waters later introduced deterministic identity-based encryption and constructed it from identity-based lossy trapdoor functions [BKPW12]. Their concrete construction maps an n -bit message to a tuple of group elements and therefore expands the ciphertext. Subsequent lattice-based constructions achieved stronger forms of adaptive security, but retained a polynomial ciphertext-expansion factor [XXZ12]. Standard CCA-secure deterministic IBE is also known, but its ciphertext concatenates a hash value with the outputs of two trapdoor functions [HYZLXY19].

There is a separate line of randomized IBE constructions described as using length-preserving symmetric encryption or as having no redundancy [LQ05]. In these schemes only the symmetric

payload is length preserving; the complete IBE ciphertext still contains a public-key header. Thus, neither this terminology nor the existence of prior deterministic IBE gives zero expansion for the complete ciphertext. Our application is, to the best of our knowledge, the first deterministic IBE with identical message and ciphertext spaces. It additionally provides a punctured decryption key while preserving weak-source pseudorandomness in the random-oracle model, under the source condition of Definition 3.4.

Identity-based unique signatures. Hovd and Stam [HS20] studied identity-based signatures with *outsider unicity* for vetted encryption. Their guarantee is computational: even given identity-specific signing keys, an efficient adversary cannot find two distinct valid signatures for the same identity–message pair. Our scheme has *perfect uniqueness*: for every honestly generated set of public parameters and fixed random oracle, each identity–message pair admits exactly one valid signature, regardless of computational power or access to signing keys. We establish adaptive unforgeability separately.

1.6 Independent and concurrent work

In independent and concurrent work, Canetti, Luo, and Zhang [CLZ26] also study applications of Shmueli–Zhandry permutable PRPs and formulate closely related notions of puncturable PRPs and trapdoor permutations. Their puncturable-PRP definition, like ours, allows puncturing forward and inverse evaluation and requires pseudorandomness of a designated image given the punctured key. They use these tools to obtain CCA-secure public-key encryption, simulation-secure functional encryption under chosen-ciphertext attacks, and unary homomorphic encryption, and give a reversible-circuit instantiation under split-circuit pseudorandomness. Our focus is on full message and key domains in the CRS model with two-level punctured security, and applications to deterministic IBE with zero ciphertext expansion, identity-based unique signatures, and direct PPAD hardness.

1.7 Paper organization

Section 2 gives an overview of our constructions and proof techniques, and Section 3 reviews the required cryptographic tools. We construct puncturable PRPs in Section 4. In Section 5, we construct TDPs with puncturable trapdoors and give their application to PPAD hardness. Section 6 constructs TDPs with full key and message domains in the CRS model and establishes two-level punctured security and weak-source pseudorandomness. Section 7 presents our deterministic identity-based encryption scheme, and Section 8 gives our identity-based unique signatures. Section 9 constructs permutable PRPs with a full key domain in the CRS model. Appendix A gives a generalization of the sparse trigger lemma used in our proofs.

AI disclosure. The ideas and proofs in this paper were developed and written by the authors. Parts of the manuscript were written with the assistance of AI tools.

2 Technical overview

We construct a TDP with a full key and message domain in the CRS model. We require *two-level punctured security*. For a uniformly random key and challenge output, inversion must remain hard even given a punctured master key that derives trapdoors for all other keys and a punctured trapdoor that inverts every other output under the challenge key. Forward evaluation remains unrestricted.

We begin in Section 2.1 by reviewing the permutable-PRP notion of Shmueli and Zhandry [SZ25], which we use as a building block. In Section 2.2, we introduce our puncturable-PRP notion, which is a main tool in our construction of TDPs with a full key domain. We then introduce TDPs with puncturable trapdoors, which have a full message domain but a sparse key domain, in Section 2.3, and construct a TDP with a full key and message domain in the CRS model in Section 2.4. We use the same key encoding to obtain permutable PRPs with a full key domain in Section 2.5. In Section 2.6, we prove puncturable-PRP security. We conclude with the deterministic-IBE and End-of-Line applications.

2.1 Permutable PRPs

Our constructions build on permutable PRPs, introduced by Shmueli and Zhandry [SZ25]. These allow us to modify the permutation defined by a secret base key by composing it with a chosen permutation on either the input or the output. The result is a derived key that supports both forward and inverse evaluation. Crucially, even a holder of this key who knows the chosen permutation cannot distinguish whether the key implements the original permutation or the modified one.

An *output-permutable PRP* extends an ordinary PRP with an algorithm that takes a base key k , a permutation Γ over $[N]$, and a bit b , and outputs a derived key

$$k^{\Gamma,b} \leftarrow \text{Permute}(k, \Gamma, b).$$

Write $\Pi(k, \cdot)$ for the base permutation and $\Pi^{-1}(k, \cdot)$ for its inverse. For $b = 0$, the derived key evaluates the original permutation. For $b = 1$, it additionally applies Γ to the output. In both cases, it also supports inverse evaluation:

$$\begin{aligned} \text{Eval}(k^{\Gamma,b}, x) &= \begin{cases} \Pi(k, x) & b = 0, \\ \Gamma(\Pi(k, x)) & b = 1, \end{cases} \\ \text{Eval}^{-1}(k^{\Gamma,b}, y) &= \begin{cases} \Pi^{-1}(k, y) & b = 0, \\ \Pi^{-1}(k, \Gamma^{-1}(y)) & b = 1. \end{cases} \end{aligned}$$

Security guarantees that the derived key hides whether Γ was applied. The adversary chooses N and Γ before the base key $k \leftarrow \{0, 1\}^\lambda$ is sampled. The two derived keys must be computationally indistinguishable, even given the chosen permutation:

$$(\Gamma, k^{\Gamma,0}) \approx (\Gamma, k^{\Gamma,1}).$$

An *input-permutable PRP* instead applies Γ and Γ^{-1} to the input when $b = 1$.

We use both notions for general swaps, namely transpositions (a, b) that exchange a and b and fix every other point. Shmueli and Zhandry give transformations between input and output permutability and a construction supporting both [SZ25]. See Definition 3.6 for the formal definition.

A useful feature of permutability security is that a reduction can receive the derived key itself, rather than only oracle access to a challenge permutation. It can therefore construct and obfuscate circuits that evaluate the permutation and its inverse without knowing the base key. We use this feature throughout our proofs.

2.2 Puncturable PRPs - definitions

A puncturable PRF preserves evaluation at every input except a fixed input x^* , while its value at x^* remains indistinguishable from random given the punctured key [SW14]. We seek a corresponding guarantee for PRPs. For a candidate image y^* , however, unrestricted inverse evaluation would reveal whether $\Pi^{-1}(k, y^*) = x^*$. The punctured key must therefore restrict forward evaluation at x^* and inverse evaluation at the candidate image y^* . The choice of which output to puncture is crucial: our definition differs from the notion ruled out by Boneh, Kim, and Wu [BKW17], as we explain below.

To randomize the value at x^* , a natural approach is to use input permutability to swap x^* with $x' \leftarrow [N]$, sampled independently of the base key. The new image of x^* is $\Pi(k, x')$, which is uniform for every fixed base key. However, the image of x' also changes to $\Pi(k, x^*)$. The inverse changes accordingly: at $\Pi(k, x^*)$, it now returns x' instead of x^* . Puncturing x^* and its new image does not remove either of these changes. We need a key that preserves the original forward and inverse evaluations outside their respective punctures, while hiding whether the candidate image is genuine or independently sampled.

Given a base key k and any pair $x^*, y^* \in [N]$, puncturing produces $k\{x^*, y^*\} \leftarrow \text{Punc}(k, x^*, y^*)$. Note that the chosen puncture points need not be related. This key preserves forward evaluation at every input except x^* and inverse evaluation at every output except y^* . At their respective puncture points, the algorithms return $\perp$. Formally,

$$\begin{aligned} \text{Eval}(k\{x^*, y^*\}, \cdot) &= \Pi(k, \cdot)[x^* \rightarrow \perp], \\ \text{Eval}^{-1}(k\{x^*, y^*\}, \cdot) &= \Pi^{-1}(k, \cdot)[y^* \rightarrow \perp]. \end{aligned}$$

Here and in the rest of the technical overview, $f[a \rightarrow v]$ stands for the function obtained from f by replacing its value at a with v .

Security compares a genuine input-output pair with a pair in which the image or preimage is replaced by an independent uniformly random value. In each experiment, the distinguisher receives both the pair and a key punctured at that pair. For every fixed input x^* , set $y^* = \Pi(k, x^*)$ and sample $U \leftarrow [N]$. Puncturable input security requires that no PPT distinguisher can distinguish the following distributions with non-negligible advantage:

$$(x^*, k\{x^*, y^*\}, y^*) \approx (x^*, k\{x^*, U\}, U).$$

For every fixed output y^* , set $x^* = \Pi^{-1}(k, y^*)$ and sample $U \leftarrow [N]$. Puncturable output security requires the analogous indistinguishability:

$$(x^*, k\{x^*, y^*\}, y^*) \approx (U, k\{U, y^*\}, y^*).$$

In both comparisons, $k \leftarrow \{0, 1\}^\lambda$ is sampled uniformly at random, and each punctured key is generated at the pair displayed in its own experiment.

The difference from the notion ruled out by Boneh, Kim, and Wu [BKW17] lies in the random experiment. In their notion, puncturing at x^* fixes the inverse puncture at $y^* = \Pi(k, x^*)$. The

punctured key remains unchanged when the challenge is replaced by a uniform value U . Consequently, inversion of the challenge succeeds in the random experiment except when $U = y^*$, but fails in the real experiment. Our input-security definition instead punctures the inverse at the displayed challenge: at y^* in the real experiment and at U in the random experiment. The challenge and the inverse puncture therefore change together, so inversion of the challenge returns $\perp$ in both experiments. Thus, we allow the forward and inverse punctures to be specified independently; they need not form an input-output pair of the underlying permutation. In particular, when $U \neq y^*$, our random experiment disables inversion at U , where their correctness requirement demands successful inversion. Their impossibility result therefore does not apply to our definition.

We use our puncturable PRP in the TDP construction below. Its construction and security proof appear in Section 2.6.

2.3 TDP with puncturable trapdoor

Our next building block is a trapdoor permutation whose trapdoor can be punctured at a chosen output. As in an ordinary TDP, a public key $\mathbf{pk}$ permits forward evaluation and a secret trapdoor $\mathbf{td}$ permits inversion. Puncturing produces $\mathbf{td}\{y^*\} \leftarrow \text{Punc}(\mathbf{td}, y^*)$, which inverts every output except y^* , where it returns $\perp$. The public key is unchanged, so forward evaluation remains available at every input, including the missing preimage.

Security requires that this missing preimage remain hard to recover even given the punctured trapdoor. More precisely, for every output y^* fixed independently of key generation, every PPT adversary given $(\mathbf{pk}, y^*, \mathbf{td}\{y^*\})$ has negligible probability of finding x^* such that $\Pi(\mathbf{pk}, x^*) = y^*$. Thus, the adversary can invert all other outputs, but cannot recover the preimage of the punctured output. The formal definition appears in Definition 5.1.

The distinction from a puncturable PRP is that forward evaluation is unrestricted. In particular, the public key lets anyone check whether a proposed preimage maps to y^* . We therefore require hardness of recovering the preimage, rather than indistinguishability of that preimage from an independent uniform point.

We obtain this primitive using the puncturable PRP introduced in Section 2.2 and indistinguishability obfuscation.

Construction. Let base be a puncturable PRP over $[N]$. Key generation samples $\mathbf{k} \leftarrow \{0, 1\}^\lambda$ and outputs

$$\mathbf{pk} \leftarrow \text{iO}(1^\lambda, \text{base}.\Pi(\mathbf{k}, \cdot)), \quad \mathbf{td} := \mathbf{k}.$$

Forward evaluation $\Pi(\mathbf{pk}, x)$ returns $\mathbf{pk}(x)$, and inversion $\Pi^{-1}(\mathbf{td}, y)$ using $\mathbf{td} = \mathbf{k}$ returns $\text{base}.\Pi^{-1}(\mathbf{k}, y)$. To puncture the trapdoor at $y^* \in [N]$, output

$$\mathbf{td}\{y^*\} \leftarrow \text{iO}(1^\lambda, \text{base}.\Pi^{-1}[y^* \rightarrow \perp](\mathbf{k}, \cdot)),$$

where the patch makes the circuit return $\perp$ when its input y equals y^* , and otherwise return $\text{base}.\Pi^{-1}(\mathbf{k}, y)$.

Puncturable trapdoor security proof. Fix an output y^* and write $x^* = \text{base}.\Pi^{-1}(\mathbf{k}, y^*)$ for its preimage. We need to prove that a PPT adversary that is given $(\mathbf{pk}, \mathbf{td}\{y^*\})$ cannot find x^* with more than negligible probability for a uniformly random $\mathbf{k}$. The proof has two steps, both carried out inside the obfuscated forward and inverse circuits.

- *Replacing the preimage with a uniformly random value.* We apply output-puncturing security inside the obfuscated circuits. First, modify the forward circuit to return y^* explicitly at x^* . We can now replace k in both circuits by $k\{x^*, y^*\}$: the forward circuit handles x^* explicitly, and the inverse circuit already returns $\perp$ at y^* .

$$\begin{aligned} \text{pk} &\leftarrow \text{iO}(1^\lambda, \text{base}.\Pi[x^* \rightarrow y^*](k\{x^*, y^*\}, \cdot)), \\ \text{td}\{y^*\} &\leftarrow \text{iO}(1^\lambda, \text{base}.\Pi^{-1}[y^* \rightarrow \perp](k\{x^*, y^*\}, \cdot)). \end{aligned}$$

Puncturing correctness preserves both functionalities, so iO hides this change. Output-puncturing security then lets us replace $x^* = \text{base}.\Pi^{-1}(k, y^*)$ by an independent uniform point $x^* \leftarrow [N]$, with the keys punctured at the new pair (x^*, y^*) . Finally, use iO to restore the original key inside both circuits, keeping their explicit branches.

$$\begin{aligned} \text{pk} &\leftarrow \text{iO}(1^\lambda, \text{base}.\Pi[x^* \rightarrow y^*](k, \cdot)), \\ \text{td}\{y^*\} &\leftarrow \text{iO}(1^\lambda, \text{base}.\Pi^{-1}[y^* \rightarrow \perp](k, \cdot)). \end{aligned}$$

This again preserves functionality, since neither circuit uses the punctured key at its excluded point.

- *Apply the sparse-trigger lemma.* The only remaining occurrence of x^* is the special branch in the obfuscated forward circuit pk . This point is uniform and independent of the base circuit and the fixed output y^* . The unpredictability guarantee of the sparse trigger lemma (Lemma A.1) therefore says that x^* is hard to recover from this obfuscation, even given the base circuit description. That description contains k , so it also suffices to generate the punctured inverse supplied to the adversary. Thus, the punctured trapdoor does not invalidate the lemma's guarantee. Transferring this bound back through the hybrids proves that the original preimage is hard to recover.

The formal construction and proof appear in Section 5. The resulting TDP has a full message domain but a sparse key domain: correctness is guaranteed for generated public keys, not for arbitrary strings. In the next subsection, we use this primitive to obtain a TDP with a full key domain in the CRS model.

2.4 TDPs with full key and message domain

We construct a TDP with a full key domain and a full message domain in the CRS model. We require *two-level punctured security*: inversion remains hard for a uniformly random key and any challenge output fixed independently of setup, even given the master key punctured at the sampled key, which permits derivation of trapdoors for every other key, and the corresponding trapdoor punctured at the challenge output.

The scheme in Definition 6.1 includes the following algorithms. Write $\ell = \ell(\lambda, \log N)$ for its key length.

- $\text{Gen}(1^\lambda, N)$ generates a common reference string crs and a master key mk .
- $\Pi(\text{crs}, K_{\text{enc}}, x)$ returns the image of $x \in [N]$ under the permutation specified by crs and $K_{\text{enc}} \in \{0, 1\}^\ell$.
- $\text{PuncMK}(\text{mk}, K_{\text{enc}}^*)$ produces $\text{mk}\{K_{\text{enc}}^*\}$, which derives the same trapdoors for every other key and rejects K_{enc}^* .

- $\text{Trapdoor}(\text{mk}, K_{\text{enc}})$ returns a trapdoor td for inverting $\Pi(\text{crs}, K_{\text{enc}}, \cdot)$.
- $\text{PuncTD}(\text{td}, y^*)$ produces $\text{td}\{y^*\}$. Inversion with this trapdoor agrees with inversion using td at every $y \neq y^*$ and rejects y^* .
- $\Pi^{-1}(\text{td}, y)$ returns the unique x satisfying $\Pi(\text{crs}, K_{\text{enc}}, x) = y$, where $\text{td} = \text{Trapdoor}(\text{mk}, K_{\text{enc}})$.

Correctness holds for any key under an honestly generated $(\text{crs}, \text{mk}) \in \text{Gen}(1^\lambda, N)$.

For two-level punctured security, fix $y^* \in [N]$ independently of setup. Sample $(\text{crs}, \text{mk}) \leftarrow \text{Gen}(1^\lambda, N)$ and a uniform $K_{\text{enc}}^* \leftarrow \{0, 1\}^\ell$. Derive the trapdoor $\text{td} \leftarrow \text{Trapdoor}(\text{mk}, K_{\text{enc}}^*)$, and generate the punctured master key and punctured trapdoor:

$$\begin{aligned} \text{mk}\{K_{\text{enc}}^*\} &\leftarrow \text{PuncMK}(\text{mk}, K_{\text{enc}}^*), \\ \text{td}\{y^*\} &\leftarrow \text{PuncTD}(\text{td}, y^*). \end{aligned}$$

Two-level punctured security guarantees that every PPT adversary has at most negligible probability of inverting y^* under K_{enc}^* , even given $\text{mk}\{K_{\text{enc}}^*\}$ and $\text{td}\{y^*\}$. The punctured master key derives full trapdoors for every other key. The punctured trapdoor inverts every other output under K_{enc}^* . Together, they allow inversion at every key and output pair, except (K_{enc}^*, y^*) , where inversion must remain hard. Formally,

$$\Pr [\Pi(\text{crs}, K_{\text{enc}}^*, \hat{x}) = y^* \mid \hat{x} \leftarrow \mathbf{A}(\text{crs}, K_{\text{enc}}^*, y^*, \text{mk}\{K_{\text{enc}}^*\}, \text{td}\{y^*\})] \leq \text{negl}(\lambda),$$

where the probability is over the sampling specified above. This also gives security for an independently sampled uniform output. See Definition 6.1 for the formal definition.

Construction. We use two building blocks for the construction: a TDP PT with a puncturable trapdoor, a sparse key domain, and message domain $[N]$, as in Section 5, and a puncturable PRP punc . The puncturable PRP encodes the TDP's key-generation randomness. Let $r = r(\lambda, \log N)$ be the number of random bits used by PT.Gen , and for every $\rho \in \{0, 1\}^r$ let

$$(\text{pk}_\rho, \text{td}_\rho) := \text{PT.Gen}(1^\lambda, N; \rho).$$

The puncturable PRP punc is over the domain $\{0, 1\}^r$, identified with $[N']$ for $N' := 2^r$. Its secret key is $\mathbf{k}_{\text{master}}$, and the encoded TDP key is

$$K_{\text{enc}} := \text{punc}.\Pi(\mathbf{k}_{\text{master}}, \rho).$$

Thus, the public key length is r . Since $\text{punc}.\Pi$ is a permutation over $\{0, 1\}^r$, every r -bit string decodes to a unique tape $\rho \in \{0, 1\}^r$ and hence to a valid key pair $(\text{pk}_\rho, \text{td}_\rho)$. Consequently, every $K_{\text{enc}} \in \{0, 1\}^r$ defines a permutation of $[N]$ under an honestly generated CRS. Thus, the construction has a full key domain: any r -bit string is a valid public key.

We first define the circuit used for public evaluation.

Public evaluation.

$C_{\text{eval}}[\mathbf{k}_{\text{master}}](K_{\text{enc}}, x) :$

1. Compute $\rho \leftarrow \text{punc}.\Pi^{-1}(\mathbf{k}_{\text{master}}, K_{\text{enc}})$.
2. Compute $(\text{pk}_\rho, \text{td}_\rho) \leftarrow \text{PT.Gen}(1^\lambda, N; \rho)$.
3. Output $\text{PT}.\Pi(\text{pk}_\rho, x)$.

Setup samples $k_{\text{master}} \leftarrow \{0, 1\}^\lambda$, sets $\text{mk} := k_{\text{master}}$, and publishes an obfuscation of $C_{\text{eval}}[k_{\text{master}}]$ as crs . Public evaluation runs this obfuscated circuit on (K_{enc}, x) . Trapdoor derivation with the full master key decodes K_{enc} to ρ , computes $(\text{pk}_\rho, \text{td}_\rho) = \text{PT.Gen}(1^\lambda, N; \rho)$, and returns td_ρ .

To puncture the master key at K_{enc}^* , we use the following derivation circuit:

Trapdoor derivation.

$C_{\text{derive}}[k_{\text{master}}][K_{\text{enc}}^* \rightarrow \perp](K_{\text{enc}})$

1. If $K_{\text{enc}} = K_{\text{enc}}^*$, output $\perp$.
2. Compute $\rho \leftarrow \text{punc}.\Pi^{-1}(k_{\text{master}}, K_{\text{enc}})$.
3. Compute $(\text{pk}_\rho, \text{td}_\rho) \leftarrow \text{PT.Gen}(1^\lambda, N; \rho)$.
4. Output td_ρ .

The punctured master key $\text{mk}\{K_{\text{enc}}^*\}$ is an obfuscation of this circuit. To derive a trapdoor using the punctured master key, evaluate $\text{mk}\{K_{\text{enc}}^*\}(K_{\text{enc}})$. This returns $\perp$ if $K_{\text{enc}} = K_{\text{enc}}^*$, and td_ρ otherwise. Trapdoor puncturing and inversion use PT.Punc and $\text{PT}.\Pi^{-1}$, respectively.

The formal construction appears in Construction 6.8.

Two-level punctured security proof. Fix the challenge output y^* independently of setup and let K_{enc}^* be the uniformly sampled encoded key. Write

$$\begin{aligned}\rho^* &:= \text{punc}.\Pi^{-1}(k_{\text{master}}, K_{\text{enc}}^*), \\ (\text{pk}^*, \text{td}^*) &:= \text{PT.Gen}(1^\lambda, N; \rho^*).\end{aligned}$$

Although ρ^* is marginally uniform, the CRS ties it to K_{enc}^* through the outer decoding. A reduction given only a base public key and its punctured trapdoor cannot reproduce this relation without knowing the key-generation randomness. We first replace ρ^* by an independent uniform tape, then rewrite the selected branch to use only pk^* .

First, use iO to introduce a branch at K_{enc}^* in C_{eval} . This branch generates the base key pair from the hardwired tape ρ^* ; all other keys are decoded normally. The change preserves functionality. The derivation circuit already rejects K_{enc}^* , so it needs no copy of ρ^* . Next, puncture the outer key at the pair

$$k_{\text{master}}^{\text{punc}} \leftarrow \text{punc.Punc}(k_{\text{master}}, \rho^*, K_{\text{enc}}^*).$$

Replace the outer key in both obfuscated circuits by $k_{\text{master}}^{\text{punc}}$, using punc.Eval^{-1} for decoding. Neither circuit decodes the selected key, and puncturing preserves decoding at every other key. Thus, iO hides both replacements.

We now apply output-puncturing security of punc to replace ρ^* by an independent uniform string in $\{0, 1\}^r$, with the outer key punctured at the corresponding pair in each experiment. For each fixed K_{enc}^* , the tuple $(\rho^*, k_{\text{master}}^{\text{punc}}, K_{\text{enc}}^*)$ suffices to generate both obfuscations, run PT.Gen to obtain $(\text{pk}^*, \text{td}^*)$, and produce $\text{td}^*\{y^*\} \leftarrow \text{PT.Punc}(\text{td}^*, y^*)$. It also suffices to check the adversary's success by evaluating $\text{PT}.\Pi(\text{pk}^*, \hat{x})$. Consequently, output-puncturing security bounds the change in inversion probability, including all the information given to the adversary. Averaging over the uniform encoded key gives the same bound.

Next, use iO to restore the unpunctured outer key in both circuits, retaining the special evaluation branch and the rejection in C_{derive} . Even though the pair $(\rho^*, K_{\text{enc}}^*)$ may now be inconsistent,

puncturing correctness preserves decoding at every other key. In particular, the derivation circuit still returns the original trapdoor at each of those keys. The punctured outer key is now absent from the view, and ρ^* is independent of both k_{master} and K_{enc}^* .

The final iO step replaces the selected branch's generation of (pk^*, td^*) from ρ^* by the hardwired public key pk^* . On input (K_{enc}^*, x) , it now simply outputs $PT.\Pi(pk^*, x)$. This is functionally equivalent, and all other inputs are unchanged. The base key pair is honestly generated, while the simulation needs only its public key and the supplied punctured trapdoor.

Given a base challenge $(pk^*, td^*\{y^*\}, y^*)$, the reduction can therefore sample k_{master} and K_{enc}^* independently, generate the two obfuscations, and pass the supplied punctured trapdoor to the adversary. This simulates the final hybrid exactly without knowing ρ^* or td^* . Any successful inversion under the selected encoded key inverts y^* under pk^* , contradicting the puncturable trapdoor security of PT. The proof uses five hybrid transitions, with six applications of iO and one application of outer output-puncturing security. Its error is at most

$$\epsilon_{PT}(\lambda, N) + 6\epsilon_{iO}(\lambda) + \epsilon_{\text{punc}}(\lambda, 2^r),$$

which is negligible for the instantiations in Theorem 6.5.

2.5 Permutable PRPs with a full key domain

We construct an output-permutable PRP with a full key domain in the CRS model. Under every honestly generated **crs**, every string of the prescribed key length specifies a permutation of $[N]$ and its inverse. A private permute key **pv** allows derivation of a key $k^{\Gamma, b}$ that implements either the base permutation, when $b = 0$, or its composition with Γ , when $b = 1$. For a uniformly random base key, we require the derived key to be indistinguishable from a uniformly random string of the same length, even when (Γ, b) is chosen after seeing the **crs**. Thus, both choices of b are hidden through a common uniform-key distribution. Together with the full key domain, this means that a key can be sampled simply by choosing a uniform string, without running the private derivation algorithm, and still look like a key derived for a chosen output permutation. The formal definition appears in Definition 9.1.

Construction and correctness. The construction uses two PRP schemes. An inner permutable PRP **base** evaluates the base permutation on the message space $[N]$. An outer puncturable PRP **punc** encodes $M = (\Gamma \parallel b \parallel k)$, where $k \in \{0, 1\}^\lambda$ is the base key, Γ describes a swap on $[N]$, and $b \in \{0, 1\}$. We use fixed-length swap descriptions, with out-of-range endpoints interpreted as the identity, so every description defines a permutation. The encoding of M is the public key $K_{\text{enc}} \leftarrow \text{punc}.\Pi(k_{\text{master}}, M)$, where k_{master} is the outer secret key. Set $\ell := \lambda + 1 + \lceil \log |G_N| \rceil$ to be the length of this tuple and its encoding. The outer PRP acts on $\{0, 1\}^\ell$, identified with $[N']$ for $N' := 2^\ell$. Every string $M \in \{0, 1\}^\ell$ defines the following permutation and inverse:

$$\begin{aligned} \Pi_M(x) &= \begin{cases} \text{base}.\Pi(k, x) & b = 0, \\ \Gamma(\text{base}.\Pi(k, x)) & b = 1, \end{cases} \\ \Pi_M^{-1}(y) &= \begin{cases} \text{base}.\Pi^{-1}(k, y) & b = 0, \\ \text{base}.\Pi^{-1}(k, \Gamma^{-1}(y)) & b = 1. \end{cases} \end{aligned}$$

The outer inverse decodes an encoded key as $M = \text{punc}.\Pi^{-1}(k_{\text{master}}, K_{\text{enc}})$. Setup samples k_{master}

and publishes a CRS containing obfuscations of the following forward and inverse evaluation circuits:

$$\begin{aligned} C_{\text{eval}}[\mathbf{k}_{\text{master}}](K_{\text{enc}}, x) &:= \Pi_{\text{punc.}\Pi^{-1}(\mathbf{k}_{\text{master}}, K_{\text{enc}})}(x), \\ C_{\text{inv}}[\mathbf{k}_{\text{master}}](K_{\text{enc}}, y) &:= \Pi_{\text{punc.}\Pi^{-1}(\mathbf{k}_{\text{master}}, K_{\text{enc}})}^{-1}(y). \end{aligned}$$

Thus, both circuits first decode K_{enc} , then evaluate Π_M or Π_M^{-1} . The private permutation key $\mathbf{pv}$ stores the encoding circuit with $\mathbf{k}_{\text{master}}$ hardwired. The permutation algorithm $\text{Permute}(\mathbf{pv}, \mathbf{k}, \Gamma, b)$ outputs $K_{\text{enc}} \leftarrow \text{punc.}\Pi(\mathbf{k}_{\text{master}}, M)$ for $M := \Gamma \parallel b \parallel \mathbf{k}$. The base evaluation algorithms are those of base , and evaluation under an encoded key runs the corresponding obfuscated CRS circuit.

Decoding a derived key recovers its tuple exactly. For $b = 0$, the forward evaluator computes $\text{base.}\Pi(\mathbf{k}, \cdot)$; for $b = 1$, it computes $\Gamma \circ \text{base.}\Pi(\mathbf{k}, \cdot)$. In either case, the inverse evaluator computes the inverse of the corresponding permutation. For an arbitrary key string, the outer permutation still decodes it to a unique tuple defining a valid permutation. Hence, correctness holds over the entire key domain. The outer PRP's puncturing algorithm is used only in the security proof. The formal construction appears in Construction 9.4.

Security proof. Security requires that a key generated by Permute from a uniformly random base key be computationally indistinguishable from an independent uniform key of the same length, jointly with the crs . The adversary may choose (Γ^*, b^*) adaptively after seeing the crs . It receives neither the base key nor the private permute key. We first prove security for (Γ^*, b^*) fixed before setup. We then obtain adaptive security by complexity leveraging: we guess the chosen permutation and bit, and absorb the guessing loss by increasing the internal security parameter and using sub-exponential security.

The main obstacle to applying the inner PRP's permutability security is the dependence of the challenge encoding K_{enc}^* on the original inner key $\mathbf{k}^*$. Indeed, this encoding decodes to $M^* = (\Gamma^* \parallel b^* \parallel \mathbf{k}^*)$. Inner permutability does not guarantee security given the original key, and a reduction given only an inner permuted key cannot form this encoding of the unknown original key. We must therefore detach K_{enc}^* from M^* before invoking inner security.

To do so, we use similar technique as we did in Section 2.4. We use iO to introduce a special branch at K_{enc}^* in each CRS circuit. These branches hardwire M^* and evaluate Π_{M^*} and its inverse. All other keys are decoded normally. We then puncture the outer key at (M^*, K_{enc}^*) , which preserves the functionality of both circuits. Outer puncturable security lets us replace K_{enc}^* by an independent uniform string while retaining M^* in the special branches. We use iO to restore the unpunctured outer key outside those branches. At this point, the outer master key and the challenge string are sampled independently of $\mathbf{k}^*$, and only the hardwired tuple in the special branches still uses it.

We replace that tuple in both branches by the inner permuted key $\mathbf{k}^{*(\Gamma^*, b^*)} \leftarrow \text{base.}\text{Permute}(\mathbf{k}^*, \Gamma^*, b^*)$. By correctness of inner permutation, evaluating forward and backward with this key computes exactly Π_{M^*} and its inverse. Thus, this rewrite preserves functionality and is justified by iO . The circuits now depend on the original inner key only through its permuted key, so inner permutability security applies. We first switch the inner permutation bit to zero. With this bit, the functionality does not depend on the permutation description, so iO lets us replace Γ^* by a uniform Γ^{rand} . Inner permutability then lets us switch the zero bit to an independent uniform bit b^{rand} . Finally, by correctness and iO , we express the special branches again using the tuple $M' = (\Gamma^{\text{rand}} \parallel b^{\text{rand}} \parallel \mathbf{k}^*)$, which is now uniform over the outer message domain and independent of the challenge string and outer master key.

It remains to restore an honestly generated crs : although K_{enc}^* is already uniform, the special branches still use M' , which need not be its actual decoding. We puncture the outer key again and

apply outer puncturable security in the reverse direction to replace the independent uniform tuple M' by $\text{punc}.\Pi^{-1}(\mathbf{k}_{\text{master}}, K_{\text{enc}}^*)$. The special branches now agree exactly with ordinary decoding. Puncturability and iO therefore let us restore the full outer key and remove both branches. The resulting distribution consists of an honestly generated crs and an independent uniform key K_{enc}^* , as required. Thus, the proof uses both input puncturable security, to replace K_{enc}^* by an independent uniform key, and output puncturable security, to replace the uniform hardwired tuple by the actual decoding of that key. The construction and full proof appear in Section 9.

2.6 Puncturable PRPs - security proof

We prove puncturable output security, which is the direction used in Section 2.4. Puncturable input security follows by a symmetric argument. Recall the definition of puncturable output security. Fix $y^* \in [N]$ independently of $\mathbf{k}$, let $x^* = \Pi^{-1}(\mathbf{k}, y^*)$, and sample a uniform $x' \leftarrow [N]$. We security guarantee is that

$$(x^*, \mathbf{k}\{x^*, y^*\}, y^*) \approx (x', \mathbf{k}\{x', y^*\}, y^*).$$

Construction. As the base PRP, we use a permutable PRP supporting both input and output permutation, as in Definition 3.7. Let $\hat{C}$ denote the obfuscation of a circuit C . In the notation below, the key and the bracketed replacement rules are part of the circuit being obfuscated. For any pair $x^*, y^* \in [N]$, the punctured key $\mathbf{k}\{x^*, y^*\}$ consists of the two obfuscated circuits

$$\hat{\Pi}(\mathbf{k}, \cdot)[x^* \rightarrow \perp], \quad \hat{\Pi}^{-1}(\mathbf{k}, \cdot)[y^* \rightarrow \perp],$$

where the left circuit outputs $\perp$ on input x^* and $\Pi(\mathbf{k}, x)$ for every $x \neq x^*$. The right circuit outputs $\perp$ on input y^* and $\Pi^{-1}(\mathbf{k}, y)$ for every $y \neq y^*$.

Puncturable security. For the construction above, the required indistinguishability takes the following form:

$$\begin{aligned} & (x^*, \hat{\Pi}(\mathbf{k}, \cdot)[x^* \rightarrow \perp], \hat{\Pi}^{-1}(\mathbf{k}, \cdot)[y^* \rightarrow \perp], y^*) \\ & \approx (x', \hat{\Pi}(\mathbf{k}, \cdot)[x' \rightarrow \perp], \hat{\Pi}^{-1}(\mathbf{k}, \cdot)[y^* \rightarrow \perp], y^*). \end{aligned}$$

We use two main tools to move from the left-hand distribution to the right-hand distribution. The first is the permutable security of the underlying PRP, which hides a swap applied inside an obfuscated program. The second is the sparse-trigger lemma, Lemma A.1. Informally, this lemma lets us replace an obfuscation of a circuit C by an obfuscation of a circuit $C[a \rightarrow b]$ that differs only at a point a , provided that a is uniformly random and independent of the circuit description.

The proof uses two value swaps of the underlying permutable PRP. The first swap replaces the true preimage of y^* with the uniform value x' , but leaves an incorrect inverse evaluation at a point y' correlated with x' . Because of this dependency, the sparse-trigger lemma cannot directly remove the incorrect evaluation. The second swap moves the remaining changes to a fresh independent point y'' , after which the sparse-trigger lemma removes the incorrect evaluation.

- *Swapping the challenge output with a fresh output.* Sample $y' \leftarrow [N]$ and set $x' = \Pi^{-1}(\mathbf{k}, y')$. Conditional on the base key, x' is uniform. It is therefore independent of the key and the initial circuit data. Therefore, the sparse-trigger lemma lets us add one rule to the forward branch:

$$(x^*, \hat{\Pi}(\mathbf{k}, \cdot)[x^* \rightarrow \perp, x' \rightarrow y^*], \hat{\Pi}^{-1}(\mathbf{k}, \cdot)[y^* \rightarrow \perp], y^*).$$

Let $\Gamma := \Gamma^{(y^*, y')}$ be the transposition exchanging y^* and y' , and apply output permutability with Γ . Its two points are chosen independently of the base key. Let $k^{\Gamma, 0} \leftarrow \text{Permute}(k, \Gamma, 0)$ denote the derived key supplied to the reduction. It computes

$$\tilde{x}^* \leftarrow \text{Eval}^{-1}(k^{\Gamma, 0}, y^*), \quad \tilde{x}' \leftarrow \text{Eval}^{-1}(k^{\Gamma, 0}, y').$$

By correctness for bit 0, these values equal x^* and x' , respectively. The reduction uses them to construct the two branches and discloses $\tilde{x}^*$. Thus, it can carry out this step without knowing k . Switching the permute bit changes that candidate from x^* to x' .

After rewriting the resulting circuit using the original key, the distribution is

$$(x', \quad \widehat{\Pi}(k, \cdot)[x' \rightarrow \perp], \quad \widehat{\Pi}^{-1}(k, \cdot)[y^* \rightarrow \perp, y' \rightarrow x^*], \quad y^*).$$

These rewritings use functional equivalence and iO, while the permute-bit switch uses permutability security. The extra forward rule made the first branch return its original value at x^* after the swap. The inverse, however, still sends y' to x^* instead of x' .

The candidate x' is now uniform, but this is not yet the desired punctured key. The remaining incorrect inverse rule is at $y' = \Pi(k, x')$. Although y' is uniform, it is determined by the base permutation and the candidate x' . It is therefore not an independent trigger for the sparse-trigger lemma, so we cannot directly remove this rule.

- *Swapping the random candidate preimage with a fresh preimage.* Sample an independent uniform $y'' \leftarrow [N]$ and modify the inverse branch to return x' there. This change is hidden by the sparse-trigger lemma, since the existing circuit and the candidate are independent of y'' . The resulting distribution is

$$(x', \quad \widehat{\Pi}(k, \cdot)[x' \rightarrow \perp], \quad \widehat{\Pi}^{-1}(k, \cdot)[y^* \rightarrow \perp, y' \rightarrow x^*, y'' \rightarrow x'], \quad y^*).$$

Because the base function is a permutation, the sampling can equivalently be written as $x', x'' \leftarrow [N]$, $y' = \Pi(k, x')$, $y'' = \Pi(k, x'')$, without changing the distribution. It makes the transposition $\Gamma^{(x', x'')}$ independent of the base key, as required for input permutability.

Use this input swap, retaining the candidate x' . After returning to the base-key representation and removing a functionally redundant inverse rule, the distribution becomes

$$(x', \quad \widehat{\Pi}(k, \cdot)[x' \rightarrow \perp, x'' \rightarrow y'], \quad \widehat{\Pi}^{-1}(k, \cdot)[y^* \rightarrow \perp, y'' \rightarrow x^*], \quad y^*).$$

The additional rule inserted before the swap ensures that the inverse now returns its correct value at y' . The unwanted changes are instead at x'' in the forward direction and at its image y'' in the inverse direction.

Both changes can now be expressed using the same independent trigger x'' . The forward branch tests its input against x'' , and the inverse branch tests $\Pi^{-1}(k, y)$ against x'' . The unmodified branches and the candidate pair (x', y^*) are independent of this trigger. The replacement values y' and x^* do not depend on it either. Thus Lemma A.1 removes the two extra rules jointly, leaving the distribution as required:

$$(x', \quad \widehat{\Pi}(k, \cdot)[x' \rightarrow \perp], \quad \widehat{\Pi}^{-1}(k, \cdot)[y^* \rightarrow \perp], \quad y^*).$$

2.7 Deterministic length-preserving IBE

In identity-based encryption (IBE), any identity string serves as a public encryption key, and an authority holding a master secret key derives the corresponding decryption key. Since our scheme is length preserving and perfectly correct, encryption must be deterministic: for each fixed identity, distinct messages must have disjoint, nonempty ciphertext supports, and equal-sized message and ciphertext spaces force each support to contain exactly one ciphertext.

For deterministic encryption, ordinary indistinguishability for arbitrary messages is impossible: an adversary can encrypt candidate messages and compare the results. We instead require that the message be sampled from a *hard* distribution fixed independently of setup, and that its encryption be computationally indistinguishable from a uniformly random ciphertext. Our notion of hardness is very general: it simply requires the existence of a PRG that remains secure when its seed is sampled from this distribution. More precisely, we use an injective, length-doubling generator whose output on such a seed is pseudorandom, as formalized in Definition 3.4.

Our TDP with a full key and message domain yields such a scheme in the random-oracle model. Our construction works as follows. Let $H : \{0, 1\}^* \rightarrow \{0, 1\}^{\ell(\lambda, \log N)}$ be a random oracle whose output length equals the dense-key length. For an identity id , set $K_{\text{id}} := H(\text{id})$. Every oracle output is a valid key, which is precisely where full key-domain correctness is used. The IBE algorithms are then

$$\begin{aligned} \text{Enc}(\text{id}, m) &:= \Pi(\text{crs}, K_{\text{id}}, m), \\ \text{SK}_{\text{id}} &:= \text{Trapdoor}(\text{mk}, K_{\text{id}}), \\ \text{Dec}(\text{SK}_{\text{id}}, c) &:= \Pi^{-1}(\text{SK}_{\text{id}}, c). \end{aligned}$$

Because $\Pi(\text{crs}, K_{\text{id}}, \cdot)$ is a permutation of $[N]$, the message and ciphertext spaces are identical. Encryption is deterministic, perfectly correct, and has no ciphertext expansion.

Security is derived from the security of the underlying TDP. However, the TDP security notions described above use a uniformly random challenge, whereas here the challenge message is sampled from a hard distribution. We therefore first need to revisit all of the above TDP primitives and prove versions of their security guarantees that hold for hard distributions. In particular, the IBE reduction uses the resulting weak-source pseudorandomness guarantee for the dense-key TDP.

The IBE security notion allows an adaptively chosen challenge identity, and the adversary obtains secret keys for other identities and the challenge identity's secret key punctured at the displayed ciphertext. In either the real or random experiment, this punctured key decrypts every other ciphertext under the challenge identity. The reduction guesses the challenge identity among the identities queried to the random oracle and programs its oracle value to be the uniform challenge key K^* from the TDP experiment. It answers secret-key queries for other identities using the punctured master key $\text{mk}\{K^*\}$, since $H(\text{id}) \neq K^*$ except with negligible collision probability. The punctured challenge decryption key is supplied by the TDP experiment itself. Thus, conditioned on a correct guess and no collisions, distinguishing the challenge encryption from a uniform ciphertext is exactly the weak-source pseudorandomness game for the TDP (Definition 6.3). This gives the desired IBE security with the polynomial loss from guessing the challenge identity.

2.8 Direct hardness of End-of-Line

An End-of-Line (EOL) instance describes a directed graph in which every vertex has in-degree and out-degree at most one, together with a designated start vertex. The task is to find another endpoint,

which may lie on a different component, or a violation of the successor and predecessor consistency conditions specified below. More formally, following Bitansky, Paneth, and Rosen [BPR15], an EOL instance is a tuple (y^*, S, P) , where $y^* \in \{0, 1\}^\lambda$ and $S, P : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda$ are circuits satisfying $P(y^*) = y^* \neq S(y^*)$. A string $w \in \{0, 1\}^\lambda$ is a valid solution if

$$P(S(w)) \neq w \quad \vee \quad (w \neq y^* \wedge S(P(w)) \neq w).$$

If the start-vertex condition is not satisfied, we also regard y^* as a valid solution.

A TDP with puncturable-trapdoor security gives the two circuits of an EOL instance almost verbatim. Sample $(pk, td) \leftarrow \text{Gen}(1^\lambda, N)$, fix an arbitrary $y_0 \in [N]$, and compute $td\{y_0\} \leftarrow \text{Punc}(td, y_0)$. Define

$$S(x) := \Pi(pk, x), \quad P(y) := \begin{cases} y_0 & y = y_0, \\ \Pi^{-1}(td\{y_0\}, y) & y \neq y_0. \end{cases}$$

Every cycle of $\Pi(pk, \cdot)$ not containing y_0 remains locally consistent. If $\Pi(pk, y_0) \neq y_0$, the cycle containing y_0 is cut at y_0 : the point y_0 is declared to be the start of the line, and the other endpoint is the unique point $x^* = \Pi^{-1}(td, y_0)$. In this case, a vertex x satisfies the displayed EOL solution condition if and only if $\Pi(pk, x) = y_0$. If instead $\Pi(pk, y_0) = y_0$, the start-vertex condition fails, and y_0 itself is a valid solution and a preimage of y_0 . Therefore every valid EOL solution reveals the missing preimage of y_0 and breaks punctured one-wayness.

The main feature of a TDP with a puncturable trapdoor is that it permits forward evaluation at every point and inverse evaluation at every output except one fixed output. This lets us construct the EOL circuits directly. Any EOL solution yields a preimage of the punctured output, so hardness follows directly from puncturable-trapdoor security.

Earlier PPAD hardness from one-way functions and iO proceeded through SINK-OF-VERIFIABLE-LINE and then a reversible-computation reduction to EOL. Our construction bypasses that intermediate problem. As noted in the related-work discussion, this direct route does not provide the additional structure needed to obtain CLS hardness. The formal reduction is in Construction 5.9 and theorem 5.10.

3 Preliminaries

Unless stated otherwise, the logarithms in this paper are base 2. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \dots, n\}$. We denote by U_n the uniform distribution over n -bit strings. We write $x \leftarrow \mathcal{D}$ for sampling according to $\mathcal{D}$, and $x \leftarrow S$ for sampling uniformly from a finite set S . We denote by $\circ$ the string concatenation operation. A function $\text{negl}: \mathbb{N} \rightarrow \mathbb{R}^+$ is *negligible* if for every constant $c > 0$, there exists an integer N_c such that $\text{negl}(n) < n^{-c}$ for all $n > N_c$.

Let U_N be a uniform distribution over set of all the permutations over $[N]$.

3.1 Indistinguishability Obfuscation

Indistinguishability obfuscation preserves a circuit's functionality while making obfuscations of equivalent circuits computationally indistinguishable.

Definition 3.1 (*T-Secure Indistinguishability Obfuscation* [BGIRSVY01]). *Let $T: \mathbb{N} \rightarrow \mathbb{N}$ be a function, and let iO be a PPT algorithm that takes as input a security parameter $\lambda \in \mathbb{N}$, a circuit size s , and a circuit C of size at most s . The iO algorithm is a T -secure indistinguishability obfuscator if it satisfies the following properties:*

- *Preserving Functionality. For every security parameter $\lambda \in \mathbb{N}$, a circuit size s , a circuit C of size at most s , and input x to the circuit,*

$$\Pr \left[\widehat{C}(x) = C(x) \mid \widehat{C} \leftarrow \text{iO}(1^\lambda, 1^s, C) \right] = 1 .$$

- *T-Security. We say that iO is T -secure if for every security parameter $\lambda \in \mathbb{N}$, every circuit size bound s , and every pair of circuits C_1, C_2 of size $|C_1|, |C_2| \leq s$ such that $C_1 \equiv C_2$, it holds that for every distinguisher $\mathbf{D}$ of size $\text{poly}(T(\lambda))$,*

$$\left| \Pr \left[\mathbf{D}(\widehat{C}_1) = 1 \mid \widehat{C}_1 \leftarrow \text{iO}(1^\lambda, 1^s, C_1) \right] - \Pr \left[\mathbf{D}(\widehat{C}_2) = 1 \mid \widehat{C}_2 \leftarrow \text{iO}(1^\lambda, 1^s, C_2) \right] \right| \leq \text{negl}(T(\lambda)) .$$

We say that an iO is secure if it is T -secure for $T(\lambda) = \lambda$.

Definition 3.2 (*Sub-Exponentially Secure Indistinguishability Obfuscation*). *An indistinguishability obfuscator iO is sub-exponentially secure if there exists a constant $0 < c < 1$ such that the scheme is T -secure for $T(\lambda) \geq 2^{\lambda^c}$.*

3.2 Pseudorandom Generators

A pseudorandom generator stretches a short uniform seed into a longer output that is computationally indistinguishable from uniform. We also define hard source distributions through an injective generator whose output on that source is pseudorandom.

Definition 3.3 (ϵ -secure pseudorandom generator). *Let $\epsilon: \mathbb{N} \rightarrow [0, 1]$ be a function. A length-doubling pseudorandom generator (PRG) is a uniformly constructible collection of deterministic circuits $\text{PRG} := \{\text{PRG}_N\}_{N \in \mathbb{N}}$ such that, for every $N \in \mathbb{N}$, $\text{PRG}_N: [N] \rightarrow [N^2]$ has size $\text{poly}(\log N)$. It is injective if every PRG_N is injective.*

We say that PRG is ϵ -secure if, for every $N \in \mathbb{N}$ and every distinguisher $\mathbf{A}$ of size $\text{polylog}(N)$,

$$\left| \Pr \left[\mathbf{A}(w) = 1 \mid \begin{array}{l} y \leftarrow [N] \\ w \leftarrow \text{PRG}_N(y) \end{array} \right] - \Pr \left[\mathbf{A}(w) = 1 \mid w \leftarrow [N^2] \right] \right| \leq \epsilon(N) .$$

We say that the PRG is polynomially secure if there exists a negligible function $\nu : \mathbb{N} \rightarrow [0, 1]$ such that $\epsilon(N) \leq \nu(\log N)$. When we refer to a secure PRG, we refer to a polynomially secure PRG.

More generally, setting $n = \lceil \log N \rceil$, the PRG is T -secure if every non-uniform probabilistic distinguisher of size $\text{poly}(T(n))$ has advantage $\text{negl}(T(n))$ in the displayed experiment. The PRG is sub-exponentially secure if it is T -secure for $T(n) = 2^{n^c}$ for some constant $0 < c < 1$.

Definition 3.4 ((ϵ, q) -hard distribution). Let $\epsilon : \mathbb{N} \rightarrow [0, 1]$ and $q : \mathbb{N} \rightarrow \mathbb{N}$ be functions, and let $\mathcal{D} := \{\mathcal{D}_N\}_{N \in \mathbb{N}}$ be an ensemble of distributions, where $\mathcal{D}_N$ is a distribution over $[N]$. We say that $\mathcal{D}$ is (ϵ, q) -hard if there exists a uniform collection of injective circuits $\mathbf{G}_{\mathcal{D}} := \{\mathbf{G}_N\}_{N \in \mathbb{N}}$ such that, for every $N \in \mathbb{N}$, $\mathbf{G}_N : [N] \rightarrow [N^2]$ has size at most $q(N)$, and it holds that

$$\mathbf{G}_N(\mathcal{D}_N) \approx_{\epsilon(N)} \mathbf{U}_{N^2} ,$$

where $\mathbf{U}_{N^2}$ is the uniform distribution over $[N^2]$. Equivalently, for every $N \in \mathbb{N}$ and every PPT distinguisher $\mathbf{A}$,

$$\left| \Pr \left[\mathbf{A}(w) = 1 \mid \begin{array}{l} y \leftarrow \mathcal{D}_N \\ w \leftarrow \mathbf{G}_N(y) \end{array} \right] - \Pr \left[\mathbf{A}(w) = 1 \mid w \leftarrow [N^2] \right] \right| \leq \epsilon(N) .$$

When the circuit-size bound is not explicit, we say that $\mathcal{D}$ is ϵ -hard if it is (ϵ, q) -hard for some q satisfying $q(N) = \text{polylog}(N)$.

3.3 Permutable PRP

We first define output and input permutability, then combine them into a single permutable-PRP notion with a shared base permutation.

Definition 3.5 (OP-PRP). Let $G := \{G_N\}_{N \in \mathbb{N}}$ be a collection where each G_N is a set of permutations over $[N]$. An output-permutable PRP (OP-PRP) for G is a tuple of algorithms

$$(\Pi, \Pi^{-1}, \text{Permute}, \text{Eval}, \text{Eval}^{-1})$$

with the following properties:

- Efficient permutations. For any key $\mathbf{k} \in \{0, 1\}^\lambda$ and any desired block size N , $\Pi(\mathbf{k}, \cdot)$ is an efficiently computable permutation on $[N]$ with $\Pi^{-1}(\mathbf{k}, \cdot)$ being its efficiently computable inverse.
- Output permuting. $\text{Permute}(\mathbf{k}, \Gamma, b)$ is a deterministic polynomial-time procedure which takes as input a key $\mathbf{k} \in \{0, 1\}^\lambda$, the circuit description of a permutation Γ in G_N , and a bit b . It outputs a permuted key $\mathbf{k}^{\Gamma, b}$.
- Output permuted correctness. For all $\lambda, N \in \mathbb{N}$, $\mathbf{k} \in \{0, 1\}^\lambda$, $b \in \{0, 1\}$, $\Gamma \in G_N$, and $x, z \in [N]$,

$$\begin{aligned} \text{Eval}(\mathbf{k}^{\Gamma, b}, x) &= \begin{cases} \Pi(\mathbf{k}, x) & \text{if } b = 0 \\ \Gamma(\Pi(\mathbf{k}, x)) & \text{if } b = 1 \end{cases} \\ \text{Eval}^{-1}(\mathbf{k}^{\Gamma, b}, z) &= \begin{cases} \Pi^{-1}(\mathbf{k}, z) & \text{if } b = 0 \\ \Pi^{-1}(\mathbf{k}, \Gamma^{-1}(z)) & \text{if } b = 1 \end{cases} \end{aligned}$$

We call $\mathbf{k}^{\Gamma, b}$ a permuted key.

- **Security.** For every two-phase non-uniform adversary $\mathbf{A} = \{(\mathbf{A}_{1,\lambda}, \mathbf{A}_{2,\lambda})\}_{\lambda \in \mathbb{N}}$, where each pair consists of probabilistic circuits of total size $\text{poly}(\lambda)$, there exists a negligible function ϵ such that

$$\Pr \left[b' = b \mid \begin{array}{l} (N, \Gamma, \text{aux}) \leftarrow \mathbf{A}_{1,\lambda}(1^\lambda) \\ k \leftarrow \{0, 1\}^\lambda \\ b \leftarrow \{0, 1\} \\ k^{\Gamma,b} \leftarrow \text{Permute}(k, \Gamma, b) \\ b' \leftarrow \mathbf{A}_{2,\lambda}(\text{aux}, k^{\Gamma,b}) \end{array} \right] \leq \frac{1}{2} + \epsilon(\lambda) ,$$

where $\mathbf{A}_{1,\lambda}$ is required to output a block size $N \in \mathbb{N}$, a description of a permutation $\Gamma \in G_N$, and an auxiliary string aux .

Definition 3.6 (IP-PRP). Let $G := \{G_N\}_{N \in \mathbb{N}}$ be as in Definition 3.5. An input-permutable PRP (IP-PRP) consists of the base forward and inverse algorithms together with algorithms $(\text{InPermute}, \text{Eval}, \text{Eval}^{-1})$. On input a base key k , a permutation $\Gamma \in G_N$, and a bit b , the algorithm InPermute outputs a key denoted $k_{\text{in}}^{\Gamma,b}$. Correctness requires that, for every $x, z \in [N]$,

$$\begin{aligned} \text{Eval}(k_{\text{in}}^{\Gamma,b}, x) &= \begin{cases} \Pi(k, x) & b = 0, \\ \Pi(k, \Gamma(x)) & b = 1, \end{cases} \\ \text{Eval}^{-1}(k_{\text{in}}^{\Gamma,b}, z) &= \begin{cases} \Pi^{-1}(k, z) & b = 0, \\ \Gamma^{-1}(\Pi^{-1}(k, z)) & b = 1. \end{cases} \end{aligned}$$

Security is the two-phase experiment from Definition 3.5, quantified over every two-phase non-uniform adversary consisting of probabilistic polynomial-size circuit families, with Permute replaced by InPermute . In particular, the first phase must output (N, Γ, aux) before the base key and the hidden bit are sampled, and the second phase receives only $(\text{aux}, k_{\text{in}}^{\Gamma,b})$. We denote its security error by $\epsilon_{\text{IPRP}}(\lambda, N)$. For the applications below it suffices that G_N contain all transpositions, with the transposition (x, x) interpreted as the identity.

Definition 3.7 (Permutable PRP (PPRP)). A permutable PRP for a class G is a tuple of algorithms $(\Pi, \Pi^{-1}, \text{Permute}, \text{InPermute}, \text{Eval}, \text{Eval}^{-1})$ such that $(\Pi, \Pi^{-1}, \text{Permute}, \text{Eval}, \text{Eval}^{-1})$ is an OP-PRP according to Definition 3.5, and $(\Pi, \Pi^{-1}, \text{InPermute}, \text{Eval}, \text{Eval}^{-1})$ is an IP-PRP according to Definition 3.6, for the same class G and the same base-key distribution. The evaluation algorithms accept derived keys from either interface; such keys may include a tag identifying their interface. Correctness and security are required separately for each interface. In addition, the base permutation must satisfy ordinary oracle PRP security: for every non-uniform probabilistic polynomial-size oracle distinguisher $\mathbf{A}$ and every efficiently computable $N = N(\lambda)$ with $\log N = \text{poly}(\lambda)$,

$$\left| \Pr_{k \leftarrow \{0,1\}^\lambda} \left[\mathbf{A}^{\Pi(k, \cdot)}(1^\lambda) = 1 \right] - \Pr_{\pi \leftarrow U_N} \left[\mathbf{A}^\pi(1^\lambda) = 1 \right] \right| \leq \text{negl}(\lambda).$$

The distinguisher receives oracle access, not the base key. No joint security guarantee is required when an adversary receives both an input-permuted and an output-permuted key derived from the same base key. We write $\epsilon_{\text{PRP}}(\lambda, N)$ for a common upper bound on the security errors of the two interfaces. For T -security, each interface must be secure against two-phase adversaries of total size $\text{poly}(T(\lambda))$, with excess success probability over $1/2$ bounded by $\text{negl}(T(\lambda))$ for all sufficiently large λ . The oracle PRP requirement must likewise hold for distinguishers of size $\text{poly}(T(\lambda))$ with advantage $\text{negl}(T(\lambda))$. Sub-exponential security means T -security for $T(\lambda) = 2^{\lambda^c}$ for some constant $0 < c < 1$.

Unless otherwise specified, a permutable PRP (PPRP) supports both input and output permutation for the class of general swaps, including the identity swap. We use OP-PRP or IP-PRP when only one direction is required.

Theorem 3.8 ([SZ25]). *Assume the existence of sub-exponentially secure indistinguishability obfuscation and a sub-exponentially secure length-doubling PRG. Then, for every efficiently computable domain-size function $N = N(\lambda)$ with $\log N = \text{poly}(\lambda)$, there exists a sub-exponentially secure permutable PRP (PPRP), as in Definition 3.7, for the class of general swaps on $[N]$.*

This follows from the quantitative bounds of [SZ25]: the PRG supplies the puncturable PRF used in the neighbor-swap construction, and iO extends it to general swaps. The latter reduction loses a factor $O(N)$, which is absorbed by a sufficiently large polynomial increase in the internal security parameter while retaining sub-exponential security. Composing their input- and output-permutable constructions gives both interfaces.

4 How to puncture PRPs

Our goal is to obtain a permutation analogue of puncturable PRFs, whose punctured keys preserve evaluation away from a designated input while the value at that input remains indistinguishable from random. For PRPs, we must also puncture inverse evaluation at the candidate image, since otherwise the inverse function would reveal whether the pair is genuine.

Toward achieving this goal, we use permutable PRPs. At first glance, it might seem sufficient to swap the designated value with a uniformly random one. However, a swap also changes forward and inverse evaluations at unpunctured points, violating the requirement that the restricted key agree with the original permutation outside its punctures. Using iO and a PRG , we hide these additional changes and obtain a restricted key that preserves the original forward and inverse evaluations outside their punctures. For every fixed input, its genuine image is indistinguishable from an independent uniform candidate, jointly with the key punctured at that input and the candidate image in each experiment.

We will use the puncturable-security guarantees established in this section in subsequent security proofs. They allow us to replace a designated image or preimage by an independent uniform value, jointly with the appropriately punctured key, analogously to the use of puncturable PRFs.

4.1 Definition

Definition 4.1 (Puncturable PRP). *Let $N \in \mathbb{N}$ be the domain size. A Puncturable Pseudorandom Permutation scheme over the domain $[N]$ is a tuple of efficient algorithms $(\Pi, \Pi^{-1}, \text{Punc}, \text{Eval}, \text{Eval}^{-1})$ satisfying the conditions:*

- **Functionality preserving:** *For every security parameter $\lambda \in \mathbb{N}$, key $\mathbf{k} \leftarrow \{0,1\}^\lambda$, and points $x^*, y^* \in [N]$, the punctured key $\mathbf{k}\{x^*, y^*\} \leftarrow \text{Punc}(\mathbf{k}, x^*, y^*)$ satisfies:*

$$\begin{aligned} \forall x \neq x^*, \quad \text{Eval}(\mathbf{k}\{x^*, y^*\}, x) &= \Pi(\mathbf{k}, x) \quad \text{and} \quad \text{Eval}(\mathbf{k}\{x^*, y^*\}, x^*) = \perp . \\ \forall y \neq y^*, \quad \text{Eval}^{-1}(\mathbf{k}\{x^*, y^*\}, y) &= \Pi^{-1}(\mathbf{k}, y) \quad \text{and} \quad \text{Eval}^{-1}(\mathbf{k}\{x^*, y^*\}, y^*) = \perp . \end{aligned}$$

- **Puncturable security (input):** *For every $x^* \in [N]$, the joint distribution of the punctured key and the image y^* is computationally indistinguishable from a distribution where y^* is chosen uniformly at random:*

$$\begin{aligned} & \left\{ (x^*, \mathbf{k}\{x^*, y^*\}, y^*) \mid \begin{array}{l} \mathbf{k} \leftarrow \{0,1\}^\lambda \\ y^* \leftarrow \Pi(\mathbf{k}, x^*) \\ \mathbf{k}\{x^*, y^*\} \leftarrow \text{Punc}(\mathbf{k}, x^*, y^*) \end{array} \right\} \\ & \approx_\epsilon \left\{ (x^*, \mathbf{k}\{x^*, y^*\}, y^*) \mid \begin{array}{l} \mathbf{k} \leftarrow \{0,1\}^\lambda \\ y^* \leftarrow [N] \\ \mathbf{k}\{x^*, y^*\} \leftarrow \text{Punc}(\mathbf{k}, x^*, y^*) \end{array} \right\} \end{aligned}$$

where $\epsilon = \epsilon(N, \lambda)$ is negligible in the security parameter.

- **Puncturable security (output):** *For every $y^* \in [N]$, the joint distribution of the punctured key and the preimage x^* is computationally indistinguishable from a distribution where x^* is chosen*

uniformly at random:

$$\left\{ (x^*, k\{x^*, y^*\}, y^*) \mid \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ x^* \leftarrow \Pi^{-1}(k, y^*) \\ k\{x^*, y^*\} \leftarrow \text{Punc}(k, x^*, y^*) \end{array} \right\} \\ \approx_\epsilon \left\{ (x^*, k\{x^*, y^*\}, y^*) \mid \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ x^* \leftarrow [N] \\ k\{x^*, y^*\} \leftarrow \text{Punc}(k, x^*, y^*) \end{array} \right\}$$

where $\epsilon = \epsilon(N, \lambda)$ is negligible in the security parameter.

4.2 Construction and proof

Theorem 4.2. Assume the existence of indistinguishability obfuscation with security ϵ_{IO} and an injective length-doubling PRG with ϵ_{PRG} security. For every efficiently computable domain-size function $N = N(\lambda)$ that is exponential in λ , and every security parameter $\lambda \in \mathbb{N}$, any permutable PRP scheme as in Definition 3.7 for the class of general swaps with security ϵ_{PRP} can be extended to a puncturable PRP scheme, where the algorithms $(\text{Punc}, \text{Eval}, \text{Eval}^{-1})$ are defined in Construction 4.4, with

$$\epsilon(\lambda, N) = O(1/N + \epsilon_{\text{PRG}}(N) + \epsilon_{\text{IO}}(\lambda) + \epsilon_{\text{PRP}}(\lambda, N)) .$$

The above theorem directly implies the following corollary.

Corollary 4.3. Assume there exists some $\epsilon > 0$ such that for every efficiently computable domain-size function $N = N(\lambda)$ that is exponential in λ , and every security parameter $\lambda \in \mathbb{N}$, there exists indistinguishability obfuscation with security $\epsilon_{\text{IO}}(\lambda) \leq 2^{-\epsilon\lambda}$, a PRG with security $\epsilon_{\text{PRG}}(N) \leq 1/N^\epsilon$, and the base permutable PRP scheme is $\epsilon_{\text{PRP}}(\lambda, N) \leq 2^{-\epsilon\lambda}$ secure, then the extended puncturable PRP scheme is $\epsilon(\lambda, N) = O(1/N^\epsilon + 2^{-\epsilon\lambda})$ secure.

To prove the theorem, we introduce the following construction.

Construction 4.4 (Puncturable PRP). Let PPRP be a permutable PRP as in Definition 3.7. We retain its base forward and inverse algorithms and define the puncturing and punctured-evaluation algorithms below. For every $\Pi_0, \Pi_1 : N \rightarrow N$, let $C_{\Pi_0, \Pi_1} : \{0, 1\} \times N \rightarrow \{0, 1\}$ be the following circuit.

$$C_{\Pi_0, \Pi_1}(b, z) := \begin{cases} \Pi_0(z) & \text{if } b = 0 \\ \Pi_1(z) & \text{if } b = 1 \end{cases} .$$

We construct $(\text{Punc}, \text{Eval}, \text{Eval}^{-1})$ as follows.

- $\text{Punc}(k, x^*, y^*)$:
 1. Set $\Pi_0 := \Pi[x^* \rightarrow \perp](k, \cdot)$, and set $\Pi_1 := \Pi^{-1}[y^* \rightarrow \perp](k, \cdot)$.
 2. Compute $\widehat{C} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1})$, and output $k\{y^*\} := \widehat{C}$.
- $\text{Eval}(k\{x^*, y^*\}, x)$:
 1. Parse $k\{x^*, y^*\}$ as $\widehat{C}$, and output $y \leftarrow \widehat{C}(0, x)$.

- $\text{Eval}^{-1}(\mathbf{k}\{x^*, y^*\}, y)$:

1. Parse $\mathbf{k}\{x^*, y^*\}$ as $\widehat{C}$, and output $x \leftarrow \widehat{C}(1, y)$.

Proof of Theorem 4.2. We prove puncturable security (output). The proof of puncturable security (input) follows by the same steps, by transposing the roles of Π and Π^{-1} .

Fix any $y^* \in [N]$. By construction,

$$\begin{aligned} & \left\{ (\mathbf{k}\{x^*, y^*\}, x^*, y^*) \left| \begin{array}{l} \mathbf{k} \leftarrow \{0, 1\}^\lambda \\ x^* \leftarrow \Pi^{-1}(\mathbf{k}, y^*) \\ \mathbf{k}\{x^*, y^*\} \leftarrow \text{Punc}(\mathbf{k}, x^*, y^*) \end{array} \right. \right\} \\ & \equiv \left\{ (\widehat{C}_0, x^*, y^*) \left| \begin{array}{l} \mathbf{k} \leftarrow \{0, 1\}^\lambda \\ x^* \leftarrow \Pi^{-1}(\mathbf{k}, y^*) \\ \Pi_0 := \Pi[x^* \rightarrow \perp](\mathbf{k}, \cdot) \\ \Pi_1 := \Pi^{-1}[y^* \rightarrow \perp](\mathbf{k}, \cdot) \\ \widehat{C}_0 \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\}. \end{aligned}$$

By Lemma A.1, we get that,

$$\begin{aligned} & \left\{ (\widehat{C}_0, x^*, y^*) \left| \begin{array}{l} \mathbf{k} \leftarrow \{0, 1\}^\lambda \\ x^* \leftarrow \Pi^{-1}(\mathbf{k}, y^*) \\ \Pi_0 := \Pi[x^* \rightarrow \perp](\mathbf{k}, \cdot) \\ \Pi_1 := \Pi^{-1}[y^* \rightarrow \perp](\mathbf{k}, \cdot) \\ \widehat{C}_0 \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\} \\ & \approx_{\epsilon' + \epsilon_{\text{iO}}(\lambda)} \left\{ (\widehat{C}_1, x^*, y^*) \left| \begin{array}{l} \mathbf{k} \leftarrow \{0, 1\}^\lambda \\ x^* \leftarrow \Pi^{-1}(\mathbf{k}, y^*) \\ y' \leftarrow [N] \\ x' \leftarrow \Pi^{-1}(\mathbf{k}, y') \\ \Pi_0 := \Pi \left[\begin{array}{l} x^* \rightarrow \perp \\ x' \rightarrow y^* \end{array} \right](\mathbf{k}, \cdot) \\ \Pi_1 := \Pi^{-1}[y^* \rightarrow \perp](\mathbf{k}, \cdot) \\ \widehat{C}_1 \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\}, \end{aligned}$$

where $\epsilon' \leq \frac{1}{N} + \epsilon_{\text{PRG}}(N) + O(\epsilon_{\text{iO}}(\lambda))$. Formally, for the circuit C_{Π_0, Π_1} , we define the sparse trigger:

$$J_{R=x'} J_{R=x'}[C_{\Pi_0, \Pi_1}, C'](b, z) := \begin{cases} C_{\Pi_0, \Pi_1}(b, z) & \text{if } R(b, z) \neq x' \\ C'(b, z) & \text{if } R(b, z) = x' \end{cases},$$

where R and C' are defined as:

$$R(b, z) := \begin{cases} z & \text{if } b = 0 \\ \perp & \text{if } b = 1 \end{cases}, \quad \text{and} \quad C'(b, z) := y^*.$$

Observe that when starting with Π_0, Π_1 as defined in the first experiment, $J_{x'}(b, z)$ is functionally equivalent to C_{Π_0, Π_1} instantiated with Π_0, Π_1 from the second experiment. Therefore, we get indistinguishability by Lemma A.1 and by the iO indistinguishability. Furthermore, Lemma A.1

applies to x' because y' is sampled uniformly at random from $[N]$, and since Π is a permutation, its preimage $x' \leftarrow \Pi^{-1}(k, y')$ is also uniformly distributed.

By the output-permute correctness of the permutatble PRP (Definition 3.5), and by the iO indistinguishability, we get that,

$$\left\{ \left(\widehat{C}_1, x^*, y^* \right) \left| \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ x^* \leftarrow \Pi^{-1}(k, y^*) \\ y' \leftarrow [N] \\ x' \leftarrow \Pi^{-1}(k, y') \\ \Pi_0 := \Pi \left[\begin{array}{c} x^* \rightarrow \perp \\ x' \rightarrow y^* \end{array} \right] (k, \cdot) \\ \Pi_1 := \Pi^{-1} [y^* \rightarrow \perp] (k, \cdot) \\ \widehat{C}_1 \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\} \\ \approx_{\epsilon_{\text{iO}}(\lambda)} \left\{ \left(\widehat{C}_2, x^*, y^* \right) \left| \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ y' \leftarrow [N] \\ \mathbf{k}^{(y^*, y'), 0} \leftarrow \text{Permute}(k, (y^*, y'), 0) \\ x^* \leftarrow \Pi^{-1}(\mathbf{k}^{(y^*, y'), 0}, y^*) \\ x' \leftarrow \Pi^{-1}(\mathbf{k}^{(y^*, y'), 0}, y') \\ \Pi_0 := \Pi \left[\begin{array}{c} x^* \rightarrow \perp \\ x' \rightarrow y^* \end{array} \right] (\mathbf{k}^{(y^*, y'), 0}, \cdot) \\ \Pi_1 := \Pi^{-1} [y^* \rightarrow \perp] (\mathbf{k}^{(y^*, y'), 0}, \cdot) \\ \widehat{C}_2 \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\} .$$

By the output-permute security,

$$\left\{ \left(\widehat{C}_2, x^*, y^* \right) \left| \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ y' \leftarrow [N] \\ \mathbf{k}^{(y^*, y'), 0} \leftarrow \text{Permute}(k, (y^*, y'), 0) \\ x^* \leftarrow \Pi^{-1}(\mathbf{k}^{(y^*, y'), 0}, y^*) \\ x' \leftarrow \Pi^{-1}(\mathbf{k}^{(y^*, y'), 0}, y') \\ \Pi_0 := \Pi \left[\begin{array}{c} x^* \rightarrow \perp \\ x' \rightarrow y^* \end{array} \right] (\mathbf{k}^{(y^*, y'), 0}, \cdot) \\ \Pi_1 := \Pi^{-1} [y^* \rightarrow \perp] (\mathbf{k}^{(y^*, y'), 0}, \cdot) \\ \widehat{C}_2 \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\} \\ \approx_{\epsilon_{\text{PRP}}(\lambda, N)} \left\{ \left(\widehat{C}_3, x^*, y^* \right) \left| \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ y' \leftarrow [N] \\ \mathbf{k}^{(y^*, y'), 1} \leftarrow \text{Permute}(k, (y^*, y'), 1) \\ x^* \leftarrow \Pi^{-1}(\mathbf{k}^{(y^*, y'), 1}, y^*) \\ x' \leftarrow \Pi^{-1}(\mathbf{k}^{(y^*, y'), 1}, y') \\ \Pi_0 := \Pi \left[\begin{array}{c} x^* \rightarrow \perp \\ x' \rightarrow y^* \end{array} \right] (\mathbf{k}^{(y^*, y'), 1}, \cdot) \\ \Pi_1 := \Pi^{-1} [y^* \rightarrow \perp] (\mathbf{k}^{(y^*, y'), 1}, \cdot) \\ \widehat{C}_3 \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\} .$$

In the next step, we switch from using the output-permuted key $\mathbf{k}^{(y^*, y'), 1}$ back to the original key k . We do so while maintaining the same functionality for C_{Π_0, Π_1} by modifying the programmed

points to Π_0 and Π_1 . Since permuting (y^*, y') shifts the preimages (x^*, x') for $x^* \leftarrow \Pi^{-1}(k, y^*)$ and $x' \leftarrow \Pi^{-1}(k, y')$, we change the programmed points in Π_0 from $[x^* \rightarrow \perp]$ to $[x' \rightarrow \perp]$ and from $[x' \rightarrow y^*]$ to $[x^* \rightarrow y^*]$, and add a programmed point in Π_1 that maps $[y' \rightarrow x^*]$ (since the point y^* was already programmed to output $\perp$, the permuted key does not alter the functionality at this point). Given that the circuits maintain the exact same functionality, the indistinguishability follows by iO security.

$$\left\{ \left(\widehat{C}_3, x^*, y^* \right) \left| \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ y' \leftarrow [N] \\ \textcolor{red}{k}^{(y^*, y'), 1} \leftarrow \text{Permute}(k, (y^*, y'), 1) \\ x^* \leftarrow \Pi^{-1}(k^{(y^*, y'), 1}, y^*) \\ x' \leftarrow \Pi^{-1}(k^{(y^*, y'), 1}, y') \\ \Pi_0 := \Pi \left[\begin{array}{c} x^* \rightarrow \perp \\ x' \rightarrow y^* \end{array} \right] (k^{(y^*, y')}, \cdot) \\ \Pi_1 := \Pi^{-1} [y^* \rightarrow \perp] (k^{(y^*, y')}, \cdot) \\ \widehat{C}_3 \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\} \\ \approx_{\epsilon_{\text{iO}}(\lambda)} \left\{ \left(\widehat{C}_4, x', y^* \right) \left| \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ y' \leftarrow [N] \\ x^* \leftarrow \Pi^{-1}(k, y^*) \\ x' \leftarrow \Pi^{-1}(k, y') \\ \Pi_0 := \Pi \left[\begin{array}{c} x' \rightarrow \perp \\ \textcolor{red}{x}^* \rightarrow \textcolor{red}{y}^* \end{array} \right] (k, \cdot) \\ \Pi_1 := \Pi^{-1} \left[\begin{array}{c} y^* \rightarrow \perp \\ \textcolor{red}{y}' \rightarrow \textcolor{red}{x}^* \end{array} \right] (k, \cdot) \\ \widehat{C}_4 \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\} .$$

Note that the explicit rule $[x^* \rightarrow y^*]$ is functionally redundant. Therefore, removing the rule is indistinguishable by iO.

$$\left\{ \left(\widehat{C}_4, x', y^* \right) \left| \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ y' \leftarrow [N] \\ x^* \leftarrow \Pi^{-1}(k, y^*) \\ x' \leftarrow \Pi^{-1}(k, y') \\ \Pi_0 := \Pi \left[\begin{array}{c} x' \rightarrow \perp \\ \textcolor{red}{x}^* \rightarrow \textcolor{red}{y}^* \end{array} \right] (k, \cdot) \\ \Pi_1 := \Pi^{-1} \left[\begin{array}{c} y^* \rightarrow \perp \\ y' \rightarrow x^* \end{array} \right] (k, \cdot) \\ \widehat{C}_4 \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\} \\ \approx_{\epsilon_{\text{iO}}(\lambda)} \left\{ \left(\widehat{C}_5, x', y^* \right) \left| \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ y' \leftarrow [N] \\ x^* \leftarrow \Pi^{-1}(k, y^*) \\ x' \leftarrow \Pi^{-1}(k, y') \\ \Pi_0 := \Pi [x' \rightarrow \perp] (k, \cdot) \\ \Pi_1 := \Pi^{-1} \left[\begin{array}{c} y^* \rightarrow \perp \\ y' \rightarrow x^* \end{array} \right] (k, \cdot) \\ \widehat{C}_5 \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\} .$$

By Lemma A.1, we get that,

$$\left\{ \left(\widehat{C}_5, x', y^\star \right) \left| \begin{array}{l} \mathbf{k} \leftarrow \{0, 1\}^\lambda \\ y' \leftarrow [N] \\ x^\star \leftarrow \Pi^{-1}(\mathbf{k}, y^\star) \\ x' \leftarrow \Pi^{-1}(\mathbf{k}, y') \\ \Pi_0 := \Pi[x' \rightarrow \perp](\mathbf{k}, \cdot) \\ \Pi_1 := \Pi^{-1} \left[\begin{array}{l} y^\star \rightarrow \perp \\ y' \rightarrow x^\star \end{array} \right](\mathbf{k}, \cdot) \\ \widehat{C}_5 \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\} \\ \approx_{\epsilon' + \epsilon_{\text{iO}}(\lambda)} \left\{ \left(\widehat{C}_6, x', y^\star \right) \left| \begin{array}{l} \mathbf{k} \leftarrow \{0, 1\}^\lambda \\ y' \leftarrow [N] \\ \textcolor{red}{y''} \leftarrow \textcolor{red}{[N]} \\ x^\star \leftarrow \Pi^{-1}(\mathbf{k}, y^\star) \\ x' \leftarrow \Pi^{-1}(\mathbf{k}, y') \\ \Pi_0 := \Pi[x' \rightarrow \perp](\mathbf{k}, \cdot) \\ \Pi_1 := \Pi^{-1} \left[\begin{array}{l} y^\star \rightarrow \perp \\ y' \rightarrow x^\star \\ \textcolor{red}{y''} \rightarrow \textcolor{red}{x'} \end{array} \right](\mathbf{k}, \cdot) \\ \widehat{C}_6 \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\},$$

where $\epsilon' \leq \frac{1}{N} + \epsilon_{\text{PRG}}(N) + O(\epsilon_{\text{iO}}(\lambda))$. Here, for the circuit C_{Π_0, Π_1} , we define the sparse trigger:

$$J_{R=y''}[C_{\Pi_0, \Pi_1}, C'](b, z) := \begin{cases} C_{\Pi_0, \Pi_1}(b, z) & \text{if } R(b, z) \neq y'' \\ C'(b, z) & \text{if } R(b, z) = y'' \end{cases},$$

where R and C' are defined as:

$$R(b, z) := \begin{cases} \perp & \text{if } b = 0 \\ z & \text{if } b = 1 \end{cases}, \quad \text{and} \quad C'(b, z) := x'.$$

Observe that when starting with Π_0, Π_1 as defined in the first experiment, $J_{R=y''}[C_{\Pi_0, \Pi_1}, C']$ is functionally equivalent to C_{Π_0, Π_1} instantiated with Π_0, Π_1 from the second experiment. Indistinguishability then follows by Lemma A.1 and by the iO security.

In the next step, we change the way we sample y', y'' . Since $\Pi(\mathbf{k}, \cdot)$ is a permutation over $[N]$, sampling the image points $y', y'' \leftarrow [N]$ uniformly at random and computing their preimages x', x'' is distributionally identical to sampling the preimages $x', x'' \leftarrow [N]$ uniformly at random and then

evaluating y', y'' .

$$\left\{ \left(\widehat{C}_6, x', y^* \right) \left| \begin{array}{l} \mathbf{k} \leftarrow \{0, 1\}^\lambda \\ \textcolor{red}{y'} \leftarrow [N] \\ \textcolor{red}{y''} \leftarrow [N] \\ x^* \leftarrow \Pi^{-1}(\mathbf{k}, y^*) \\ \textcolor{red}{x'} \leftarrow \Pi^{-1}(\mathbf{k}, y') \\ \Pi_0 := \Pi[x' \rightarrow \perp](\mathbf{k}, \cdot) \\ \Pi_1 := \Pi^{-1} \begin{bmatrix} y^* \rightarrow \perp \\ y' \rightarrow x^* \\ y'' \rightarrow x' \end{bmatrix} (\mathbf{k}, \cdot) \\ \widehat{C}_6 \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\} \\ \equiv \left\{ \left(\widehat{C}_7, x', y^* \right) \left| \begin{array}{l} \mathbf{k} \leftarrow \{0, 1\}^\lambda \\ \textcolor{red}{x'} \leftarrow [N] \\ \textcolor{red}{x''} \leftarrow [N] \\ x^* \leftarrow \Pi^{-1}(\mathbf{k}, y^*) \\ \textcolor{red}{y'} \leftarrow \Pi(\mathbf{k}, x') \\ \textcolor{red}{y''} \leftarrow \Pi(\mathbf{k}, x'') \\ \Pi_0 := \Pi[x' \rightarrow \perp](\mathbf{k}, \cdot) \\ \Pi_1 := \Pi^{-1} \begin{bmatrix} y^* \rightarrow \perp \\ y' \rightarrow x^* \\ y'' \rightarrow x' \end{bmatrix} (\mathbf{k}, \cdot) \\ \widehat{C}_7 \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\} .$$

By the input-permute correctness of the permutatble PRP (Definition 3.5), and by the iO indistin-

guishability, we get that,

$$\begin{aligned}
& \left\{ \left(\widehat{C}_7, x', y^* \right) \left| \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ x' \leftarrow [N] \\ x'' \leftarrow [N] \\ x^* \leftarrow \Pi^{-1}(k, y^*) \\ y' \leftarrow \Pi(k, x') \\ y'' \leftarrow \Pi(k, x'') \\ \Pi_0 := \Pi[x' \rightarrow \perp](k, \cdot) \\ \Pi_1 := \Pi^{-1} \begin{bmatrix} y^* \rightarrow \perp \\ y' \rightarrow x^* \\ y'' \rightarrow x' \end{bmatrix} (k, \cdot) \\ \widehat{C}_7 \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\} \\
& \approx_{\epsilon_{\text{iO}}(\lambda)} \left\{ \left(\widehat{C}_8, x', y^* \right) \left| \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ x' \leftarrow [N] \\ x'' \leftarrow [N] \\ \textcolor{red}{k_{\text{in}}^{(x', x''), 0}} \leftarrow \textcolor{red}{\text{InPermute}(k, (x', x''), 0)} \\ x^* \leftarrow \Pi^{-1}(k_{\text{in}}^{(x', x''), 0}, y^*) \\ y' \leftarrow \Pi(k_{\text{in}}^{(x', x''), 0}, x') \\ y'' \leftarrow \Pi(k_{\text{in}}^{(x', x''), 0}, x'') \\ \Pi_0 := \Pi[x' \rightarrow \perp](k_{\text{in}}^{(x', x''), 0}, \cdot) \\ \Pi_1 := \Pi^{-1} \begin{bmatrix} y^* \rightarrow \perp \\ y' \rightarrow x^* \\ y'' \rightarrow x' \end{bmatrix} (k_{\text{in}}^{(x', x''), 0}, \cdot) \\ \widehat{C}_8 \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\}.
\end{aligned}$$

By input-permutability security (Definition 3.6),

$$\approx_{\epsilon_{\text{PRP}}(\lambda, N)} \left\{ \left(\widehat{C}_8, x', y^* \right) \left\{ \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ x' \leftarrow [N] \\ x'' \leftarrow [N] \\ \textcolor{red}{k}_{\text{in}}^{(x', x''), 0} \leftarrow \text{InPermute}(k, (x', x''), 0) \\ x^* \leftarrow \Pi^{-1}(k_{\text{in}}^{(x', x''), 0}, y^*) \\ y' \leftarrow \Pi(k_{\text{in}}^{(x', x''), 0}, x') \\ y'' \leftarrow \Pi(k_{\text{in}}^{(x', x''), 0}, x'') \\ \Pi_0 := \Pi[x' \rightarrow \perp](k_{\text{in}}^{(x', x''), 0}, \cdot) \\ \Pi_1 := \Pi^{-1} \begin{bmatrix} y^* \rightarrow \perp \\ y' \rightarrow x^* \\ y'' \rightarrow x' \end{bmatrix} (k_{\text{in}}^{(x', x''), 0}, \cdot) \\ \widehat{C}_8 \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\} .$$

$$\left\{ \left(\widehat{C}_9, x', y^* \right) \left\{ \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ x' \leftarrow [N] \\ x'' \leftarrow [N] \\ \textcolor{red}{k}_{\text{in}}^{(x', x''), 1} \leftarrow \text{InPermute}(k, (x', x''), 1) \\ x^* \leftarrow \Pi^{-1}(k_{\text{in}}^{(x', x''), 1}, y^*) \\ y' \leftarrow \Pi(k_{\text{in}}^{(x', x''), 1}, x') \\ y'' \leftarrow \Pi(k_{\text{in}}^{(x', x''), 1}, x'') \\ \Pi_0 := \Pi[x' \rightarrow \perp](k_{\text{in}}^{(x', x''), 1}, \cdot) \\ \Pi_1 := \Pi^{-1} \begin{bmatrix} y^* \rightarrow \perp \\ y' \rightarrow x^* \\ y'' \rightarrow x' \end{bmatrix} (k_{\text{in}}^{(x', x''), 1}, \cdot) \\ \widehat{C}_9 \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\} .$$

In the next step, we switch from using the input-permuted key $k_{\text{in}}^{(x', x''), 1}$ back to the original key k . We do so while maintaining the same functionality for C_{Π_0, Π_1} by modifying the programmed points to Π_0 and Π_1 . Since permuting (x', x'') shifts the values (y', y'') for $y' \leftarrow \Pi(k, x')$ and $y'' \leftarrow \Pi(k, x'')$, we add a programmed point to Π_0 that maps $[x'' \rightarrow y']$ (since the point x' was already programmed to output $\perp$, the permuted key does not alter the functionality at this point), and modify the programmed points in Π_1 to map $[y'' \rightarrow x^*]$ instead of $[y' \rightarrow x^*]$ and map $[y' \rightarrow x']$ instead of $[y'' \rightarrow x']$. Note that these rules maintain the functionality of the permuted key if and only if for $x^* \leftarrow \Pi^{-1}(k, y^*)$, it holds that $x^* \notin \{x', x''\}$. Since x', x'' are sampled uniformly at random, this condition holds with probability $1 - 2/N$. Because the circuits maintain the exact same functionality

except with probability $2/N$, the following indistinguishability follows by iO security.

$$\approx_{\epsilon_{\text{iO}}(\lambda) + \frac{2}{N}} \left\{ \left(\widehat{C}_9, x', y^* \right) \left| \begin{array}{l} \mathbf{k} \leftarrow \{0, 1\}^\lambda \\ x' \leftarrow [N] \\ x'' \leftarrow [N] \\ \mathbf{k}_{\text{in}}^{(x', x''), 1} \leftarrow \text{InPermute}(\mathbf{k}, (x', x''), 1) \\ x^* \leftarrow \Pi^{-1}(\mathbf{k}_{\text{in}}^{(x', x''), 1}, y^*) \\ y' \leftarrow \Pi(\mathbf{k}_{\text{in}}^{(x', x''), 1}, x') \\ y'' \leftarrow \Pi(\mathbf{k}_{\text{in}}^{(x', x''), 1}, x'') \\ \Pi_0 := \Pi[x' \rightarrow \perp](\mathbf{k}_{\text{in}}^{(x', x''), 1}, \cdot) \\ \Pi_1 := \Pi^{-1} \left[\begin{array}{l} y^* \rightarrow \perp \\ y' \rightarrow x^* \\ y'' \rightarrow x' \end{array} \right] (\mathbf{k}_{\text{in}}^{(x', x''), 1}, \cdot) \\ \widehat{C}_9 \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\} .$$

$$\left\{ \left(\widehat{C}_{10}, x', y^* \right) \left| \begin{array}{l} \mathbf{k} \leftarrow \{0, 1\}^\lambda \\ x' \leftarrow [N] \\ x'' \leftarrow [N] \\ x^* \leftarrow \Pi^{-1}(\mathbf{k}, y^*) \\ y' \leftarrow \Pi(\mathbf{k}, x') \\ y'' \leftarrow \Pi(\mathbf{k}, x'') \\ \Pi_0 := \Pi \left[\begin{array}{l} x' \rightarrow \perp \\ x'' \rightarrow y' \end{array} \right] (\mathbf{k}, \cdot) \\ \Pi_1 := \Pi^{-1} \left[\begin{array}{l} y^* \rightarrow \perp \\ y'' \rightarrow x^* \\ y' \rightarrow x' \end{array} \right] (\mathbf{k}, \cdot) \\ \widehat{C}_{10} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\} .$$

Note that the explicit rule $[y' \rightarrow x']$ is functionally redundant. Therefore, removing the rule is

indistinguishable by iO security.

$$\approx_{\epsilon_{\text{iO}}(\lambda)} \left\{ \left(\widehat{C_{10}}, x', y^* \right) \left| \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ x' \leftarrow [N] \\ x'' \leftarrow [N] \\ x^* \leftarrow \Pi^{-1}(k, y^*) \\ y' \leftarrow \Pi(k, x') \\ y'' \leftarrow \Pi(k, x'') \\ \Pi_0 := \Pi \left[\begin{array}{c} x' \rightarrow \perp \\ x'' \rightarrow y' \end{array} \right] (k, \cdot) \\ \Pi_1 := \Pi^{-1} \left[\begin{array}{c} y^* \rightarrow \perp \\ y'' \rightarrow x^* \\ \textcolor{red}{y' \rightarrow x'} \end{array} \right] (k, \cdot) \\ \widehat{C_{10}} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\}.$$

By applying Lemma A.1, we get,

$$\approx_{\epsilon' + \epsilon_{\text{iO}}(\lambda)} \left\{ \left(\widehat{C_{11}}, x', y^* \right) \left| \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ x' \leftarrow [N] \\ \textcolor{red}{x'' \leftarrow [N]} \\ x^* \leftarrow \Pi^{-1}(k, y^*) \\ y' \leftarrow \Pi(k, x') \\ \textcolor{red}{y'' \leftarrow \Pi(k, x'')} \\ \Pi_0 := \Pi \left[\begin{array}{c} x' \rightarrow \perp \\ \textcolor{red}{x'' \rightarrow y'} \end{array} \right] (k, \cdot) \\ \Pi_1 := \Pi^{-1} \left[\begin{array}{c} y^* \rightarrow \perp \\ \textcolor{red}{y'' \rightarrow x^*} \end{array} \right] (k, \cdot) \\ \widehat{C_{11}} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\} \\ \approx_{\epsilon' + \epsilon_{\text{iO}}(\lambda)} \left\{ \left(\widehat{C_{12}}, x', y^* \right) \left| \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ x' \leftarrow [N] \\ \Pi_0 := \Pi \left[\begin{array}{c} x' \rightarrow \perp \end{array} \right] (k, \cdot) \\ \Pi_1 := \Pi^{-1} \left[\begin{array}{c} y^* \rightarrow \perp \end{array} \right] (k, \cdot) \\ \widehat{C_{12}} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\},$$

where $\epsilon' \leq \frac{1}{N} + \epsilon_{\text{PRG}}(N) + O(\epsilon_{\text{iO}}(\lambda))$. Here, for the circuit C_{Π_0, Π_1} , we define the sparse trigger:

$$J_{R=y''}[C_{\Pi_0, \Pi_1}, C'](b, z) := \begin{cases} C_{\Pi_0, \Pi_1}(b, z) & \text{if } R(b, z) \neq y'' \\ C'(b, z) & \text{if } R(b, z) = y'' \end{cases},$$

where R and C' are defined as:

$$R(b, z) := \begin{cases} z & \text{if } b = 0 \\ \Pi^{-1}(k, z) & \text{if } b = 1 \end{cases}, \quad \text{and} \quad C'(b, z) := \begin{cases} y' & \text{if } b = 0 \\ x^* & \text{if } b = 1 \end{cases}.$$

Observe that when starting with Π_0, Π_1 as defined in the second experiment, $J_{x''}(b, z)$ is functionally equivalent to C_{Π_0, Π_1} instantiated with Π_0, Π_1 from the first experiment. Indistinguishability then follows by Lemma A.1 and by the iO security.

By construction, we get that,

$$\begin{aligned} & \left\{ \left(\widehat{C_{12}}, x', y^* \right) \left| \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ x' \leftarrow [N] \\ \Pi_0 := \Pi \left[x' \rightarrow \perp \right] (k, \cdot) \\ \Pi_1 := \Pi^{-1} \left[y^* \rightarrow \perp \right] (k, \cdot) \\ \widehat{C_{12}} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\Pi_0, \Pi_1}) \end{array} \right. \right\} \\ & \equiv \left\{ (k\{x', y^*\}, x', y^*) \left| \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ x' \leftarrow [N] \\ k\{x', y^*\} \leftarrow \text{Punc}(k, x', y^*) \end{array} \right. \right\}, \end{aligned}$$

Overall, we get that,

$$\begin{aligned} & \left\{ (k\{x^*, y^*\}, x^*, y^*) \left| \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ x^* \leftarrow \Pi^{-1}(k, y^*) \\ k\{x^*, y^*\} \leftarrow \text{Punc}(k, x^*, y^*) \end{array} \right. \right\} \\ & \approx_\epsilon \left\{ (k\{x', y^*\}, x', y^*) \left| \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ x' \leftarrow [N] \\ k\{x', y^*\} \leftarrow \text{Punc}(k, x', y^*) \end{array} \right. \right\}, \end{aligned}$$

where $\epsilon := O(1/N + \epsilon_{\text{PRG}}(N) + \epsilon_{\text{iO}}(\lambda) + \epsilon_{\text{PRP}}(\lambda, N))$, which concludes the proof. $\square$

5 TDP with puncturable-trapdoor

In this section, we extend the TDP construction and security proof of Shmueli and Zhandry [SZ25] to obtain a TDP with a puncturable trapdoor. Their TDP provides public forward evaluation and a secret trapdoor for inversion. Our notion additionally allows the trapdoor to be punctured at an output y^* , preserving inversion at every other output while keeping inversion of y^* hard even given the punctured trapdoor. Public forward evaluation remains unrestricted. We define this notion, present the construction, and prove its security by extending their argument to account for the punctured trapdoor. We also establish weak-source security: recovering a preimage remains hard when it is sampled from a hard distribution, as defined in Definition 3.4, independently of key generation, even given its image and the trapdoor punctured at that image.

5.1 Definition

Definition 5.1 (Trapdoor Permutation (TDP) with a puncturable trapdoor). *Let $\epsilon : \mathbb{N} \times \mathbb{N} \rightarrow [0, 1]$ be a security-error function. An ϵ -secure Trapdoor Permutation (TDP) with a puncturable trapdoor is a tuple of algorithms $(\text{Gen}, \Pi, \Pi^{-1}, \text{Punc})$ with the following properties:*

- **Setup.** *The Gen algorithm is a PPT algorithm that takes as input a security parameter 1^λ and a domain size N , where $\lambda, N \in \mathbb{N}$, and outputs a public key pk and a trapdoor td .*
- **Efficient permutations.** *The algorithm Π is a deterministic polynomial-time algorithm that given the key $(\text{pk}, \cdot) \in \text{Gen}(1^\lambda, N)$ computes a permutation over $[N]$.*
- **Trapdoor correctness.** *For every $\lambda, N \in \mathbb{N}$, and for every $y \in [N]$,*

$$\Pr \left[\Pi(\text{pk}, \Pi^{-1}(\text{td}, y)) = y \mid (\text{pk}, \text{td}) \leftarrow \text{Gen}(1^\lambda, N) \right] = 1 .$$

- **Puncturable trapdoor correctness.** *For every $\lambda, N \in \mathbb{N}$, and for every $y, y' \in [N]$ where $y \neq y'$,*

$$\Pr \left[\Pi(\text{pk}, \Pi^{-1}(\text{td}\{y'\}, y)) = y \mid \begin{array}{l} (\text{pk}, \text{td}) \leftarrow \text{Gen}(1^\lambda, N) \\ \text{td}\{y'\} \leftarrow \text{Punc}(\text{td}, y') \end{array} \right] = 1 .$$

In addition, for the above experiment, it holds that $\Pi^{-1}(\text{td}\{y'\}, y') = \perp$.

- **Puncturable trapdoor security.** *For every $\lambda, N \in \mathbb{N}$, for every $y \in [N]$, and for every PPT adversary $\mathbf{A}$,*

$$\Pr \left[\Pi(\text{pk}, x) = y \mid \begin{array}{l} (\text{pk}, \text{td}) \leftarrow \text{Gen}(1^\lambda, N) \\ \text{td}\{y\} \leftarrow \text{Punc}(\text{td}, y) \\ x \leftarrow \mathbf{A}(\text{pk}, \text{td}\{y\}, y) \end{array} \right] \leq \epsilon(\lambda, N) .$$

- **Puncturable trapdoor hiding.** *For every $\lambda, N \in \mathbb{N}$, and for every PPT adversary $\mathbf{A}$,*

$$\Pr \left[y' = y \mid \begin{array}{l} (\text{pk}, \text{td}) \leftarrow \text{Gen}(1^\lambda, N) \\ y \leftarrow [N] \\ \text{td}\{y\} \leftarrow \text{Punc}(\text{td}, y) \\ y' \leftarrow \mathbf{A}(\text{pk}, \text{td}, \text{td}\{y\}) \end{array} \right] \leq \epsilon(\lambda, N) .$$

Definition 5.2 (Weak-source puncturable one-wayness). *A TDP with a puncturable trapdoor is ϵ_{src} -one-way if for every collection of distributions $\mathcal{D} := \{\mathcal{D}_N\}_{N \in \mathbb{N}}$ that is $\epsilon_{\mathcal{D}}$ -hard, for every PPT adversary $\mathbf{A}$ and every security parameter $\lambda \in \mathbb{N}$,*

$$\Pr \left[\hat{x} = x \mid \begin{array}{l} x \leftarrow \mathcal{D}_N \\ (\text{pk}, \text{td}) \leftarrow \text{Gen}(1^\lambda, N) \\ y \leftarrow \Pi(\text{pk}, x) \\ \text{td}\{y\} \leftarrow \text{Punc}(\text{td}, y) \\ \hat{x} \leftarrow \mathbf{A}(\text{pk}, \text{td}\{y\}, y) \end{array} \right] \leq \epsilon_{\text{src}}(\lambda, N, \epsilon_{\mathcal{D}}) .$$

Definition 5.3 (Weak-source pseudorandomness). *Fix size bound $q(\cdot) = \text{polylog}(\cdot)$. A TDP with a puncturable trapdoor satisfies $\epsilon_{\text{src-pr}}$ -weak-source pseudorandomness if the following holds. For every $(\epsilon_{\mathcal{D}}, q)$ -hard source ensemble $\mathcal{D} = \{\mathcal{D}_N\}_{N \in \mathbb{N}}$ fixed independently of key generation, every non-uniform PPT distinguisher $\mathbf{A}$, and every efficiently computable domain-size function $N = N(\lambda)$ that is exponential in λ , the two experiments below satisfy*

$$|\delta_{\text{real}} - \delta_{\text{rand}}| \leq \epsilon_{\text{src-pr}}(\lambda, N, \epsilon_{\mathcal{D}}),$$

where

$$\begin{aligned} \delta_{\text{real}} &:= \Pr \left[\mathbf{A}(\text{pk}, \text{td}\{y^*\}, y^*) = 1 \mid \begin{array}{l} x^* \leftarrow \mathcal{D}_N \\ (\text{pk}, \text{td}) \leftarrow \text{Gen}(1^\lambda, N) \\ y^* \leftarrow \Pi(\text{pk}, x^*) \\ \text{td}\{y^*\} \leftarrow \text{Punc}(\text{td}, y^*) \end{array} \right], \\ \delta_{\text{rand}} &:= \Pr \left[\mathbf{A}(\text{pk}, \text{td}\{y^*\}, y^*) = 1 \mid \begin{array}{l} (\text{pk}, \text{td}) \leftarrow \text{Gen}(1^\lambda, N) \\ y^* \leftarrow [N] \\ \text{td}\{y^*\} \leftarrow \text{Punc}(\text{td}, y^*) \end{array} \right]. \end{aligned}$$

Theorem 5.4. *Assume the existence of a secure indistinguishability obfuscation, an injective length-doubling pseudorandom generator, and a secure PPRP over $[N]$ for the class of general swaps. Then, there exists an ϵ -secure TDP over $[N]$ with a puncturable trapdoor, such that for every efficiently computable domain-size function $N = N(\lambda)$ that is exponential in λ , and every security parameter $\lambda \in \mathbb{N}$, it holds that, $\epsilon(\lambda, N) \leq \text{negl}(\lambda)$.*

Theorem 5.5 (Weak-source pseudorandomness). *Fix $q(\cdot) = \text{polylog}(\cdot)$. Assume the existence of secure indistinguishability obfuscation, a secure injective length-doubling PRG, and a secure PPRP for the class of general swaps. Then Construction 5.7, instantiated with this PPRP, satisfies Definition 5.3. More precisely, for every $(\epsilon_{\mathcal{D}}, q)$ -hard source ensemble $\mathcal{D}$, every efficiently computable domain-size function $N = N(\lambda)$ that is exponential in λ , and every security parameter $\lambda \in \mathbb{N}$,*

$$\epsilon_{\text{src-pr}}(\lambda, N, \epsilon_{\mathcal{D}}) \leq O(\epsilon_{\mathcal{D}}(N)) + \text{negl}(\lambda).$$

The running time of the scheme is $\text{poly}(\lambda, \log N, q(N))$.

Corollary 5.6 (Weak-source security). *Fix $q(\cdot) = \text{polylog}(\cdot)$. Assume there exists a TDP with a puncturable trapdoor over $[N]$ that satisfies puncturable trapdoor security with error $\epsilon(\lambda, N)$, as in Definition 5.1, and weak-source pseudorandomness with error $\epsilon_{\text{src-pr}}$, as in Definition 5.3. Then the same TDP satisfies weak-source puncturable one-wayness as defined in Definition 5.2. For every $(\epsilon_{\mathcal{D}}, q)$ -hard source ensemble $\mathcal{D}$ independent of key generation and every efficiently computable domain-size function $N = N(\lambda)$ that is exponential in λ , its security error satisfies*

$$\epsilon_{\text{src}}(\lambda, N, \epsilon_{\mathcal{D}}) \leq \epsilon_{\text{src-pr}}(\lambda, N, \epsilon_{\mathcal{D}}) + \epsilon(\lambda, N).$$

In particular, assume the existence of secure indistinguishability obfuscation, a secure PPRP for general swaps, and a secure injective length-doubling PRG. Then

$$\epsilon_{\text{src}}(\lambda, N, \epsilon_{\mathcal{D}}) \leq O(\epsilon_{\mathcal{D}}(N)) + \text{negl}(\lambda).$$

5.2 Construction

Let $\lambda, N \in \mathbb{N}$ denote the security parameter and domain-size parameter of the scheme. Let $q(\cdot) = \text{polylog}(\cdot)$ be a bound on the size of the PRG circuits witnessing $(\cdot, q)$ -hardness as defined in Definition 3.4. The following construction is used for Theorems 5.4 and 5.5 and corollary 5.6.

Let $(\text{base}.\Pi, \text{base}.\Pi^{-1}, \text{base}.\text{Permute}, \text{base}.\text{Eval}, \text{base}.\text{Eval}^{-1})$ be the corresponding permutable PRP scheme over $[N]$ for the class of general swaps G , and let iO be an indistinguishability obfuscation algorithm.

We define two circuit families parameterized by the PPRP key $k \in \{0, 1\}^\lambda$:

$C_{\text{eval}}[k](x)$:

1. Output $\text{base}.\Pi(k, x)$.

$C_{\text{inv}}[k][y^* \rightarrow \perp](y)$:

1. If $y = y^*$, output $\perp$.
2. Else output $\text{base}.\Pi^{-1}(k, y)$.

Construction 5.7. The TDP scheme with a puncturable trapdoor is defined as follows.

- $\text{Gen}(1^\lambda, N)$:
 1. Set $s := s(\lambda, \log N)$ for a fixed polynomial s that upper bounds the size of the circuit in all the hybrids in the proof.
 2. Sample $k \leftarrow \{0, 1\}^\lambda$.
 3. Compute $\widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{eval}}[k])$.
 4. Output (pk, td) , where $\text{pk} := \widehat{C_{\text{eval}}}$ and $\text{td} := k$.
- $\Pi(\text{pk}, x)$:
 1. Parse pk as $\widehat{C_{\text{eval}}}$.
 2. Output $\widehat{C_{\text{eval}}}(x)$.
- $\Pi^{-1}(\text{td}, y)$:
 1. Parse td as k .
 2. Output $\text{base}.\Pi^{-1}(k, y)$.
- $\text{Punc}(\text{td}, y^*)$:
 1. Parse td as k .
 2. Compute $\widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{inv}}[k][y^* \rightarrow \perp])$.
 3. Output $\text{td}\{y^*\} := \widehat{C_{\text{inv}}}$.

5.3 Proof of Theorem 5.4

Proof. Correctness follows from the permutation property of base and correctness of iO . We prove puncturable trapdoor security and then punctured trapdoor hiding.

Puncturable trapdoor security. Fix $y^* \in [N]$ independently of key generation and a PPT inverter $\mathbf{A}$. Apply Theorem 4.2 to the underlying PPRP base and write punc for its puncturable extension, which retains the same base forward and inverse algorithms. Let $\epsilon_{\text{iO}}(\lambda)$ and $\epsilon_{\text{punc}}(\lambda, N)$

denote the security errors of the indistinguishability obfuscator and the puncturable PRP respectively. Write $\epsilon_{\text{PRG}}(N)$ and $\epsilon_{\text{PPRP}}(\lambda, N)$ for the underlying PRG and PPRP errors. Pad all compared circuits to a common sufficiently large polynomial size s , including the circuits used in Lemma A.1.

The proof follows the puncture key, randomize value, and then restore key sequence. Then, to prove unpredictability of x^* , we use the unpredictability property of Lemma A.1. For each hybrid, let p_i denote the probability that $\mathbf{A}$ outputs x^* .

Hybrid 0: The real experiment. In this hybrid, we use the following base circuits:

$C_{\text{eval}}[\mathbf{k}](x)$:

1. Output $\text{base}.\Pi(\mathbf{k}, x)$.

$C_{\text{inv}}[\mathbf{k}, y^*](y)$:

1. If $y = y^*$, output $\perp$.
2. Otherwise, output $\text{base}.\Pi^{-1}(\mathbf{k}, y)$.

The success probability in this hybrid is

$$p_0 := \Pr \left[\mathbf{A}(\text{pk}^*, \text{td}^*\{y^*\}, y^*) = x^* \mid \begin{array}{l} \mathbf{k} \leftarrow \{0, 1\}^\lambda \\ x^* \leftarrow \text{base}.\Pi^{-1}(\mathbf{k}, y^*) \\ \text{pk}^* \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{eval}}[\mathbf{k}]) \\ \text{td}^*\{y^*\} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{inv}}[\mathbf{k}, y^*]) \end{array} \right].$$

In this experiment, x^* is the unique preimage of y^* , so p_0 is exactly the adversary's success probability in Definition 5.1.

Hybrid 1: Puncturing the base key. Use the following circuits that includes the punctured key $\mathbf{k}\{x^*, y^*\} \leftarrow \text{punc}.\text{Punc}(\mathbf{k}, x^*, y^*)$ and the values of the punctured points.

$C_{\text{eval}}[\mathbf{k}\{x^*, y^*\}, x^*, y^*](x)$:

1. If $x = x^*$, output y^* .
2. Otherwise, output $\text{punc}.\text{Eval}(\mathbf{k}\{x^*, y^*\}, x)$.

$C_{\text{inv}}[\mathbf{k}\{x^*, y^*\}, y^*](y)$:

1. If $y = y^*$, output $\perp$.
2. Otherwise, output $\text{punc}.\text{Eval}^{-1}(\mathbf{k}\{x^*, y^*\}, y)$.

The experiment is

$$p_1 := \Pr \left[\mathbf{A}(\text{pk}^*, \text{td}^*\{y^*\}, y^*) = x^* \mid \begin{array}{l} \mathbf{k} \leftarrow \{0, 1\}^\lambda \\ x^* \leftarrow \text{base}.\Pi^{-1}(\mathbf{k}, y^*) \\ \mathbf{k}\{x^*, y^*\} \leftarrow \text{punc}.\text{Punc}(\mathbf{k}, x^*, y^*) \\ \text{pk}^* \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{eval}}[\mathbf{k}\{x^*, y^*\}, x^*, y^*]) \\ \text{td}^*\{y^*\} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{inv}}[\mathbf{k}\{x^*, y^*\}, y^*]) \end{array} \right].$$

Puncturing correctness makes both circuits functionally equivalent to those of Hybrid 0, including at their explicitly handled points. Hence,

$$|p_1 - p_0| \leq 2\epsilon_{\text{iO}}(\lambda).$$

Hybrid 2: Randomizing the designated preimage. Here, we keep the same circuits, but change the way we sample x^* . By output-puncturing security of `punc`, we can replace $x^* = \text{base}.\Pi^{-1}(k, y^*)$ by an independent uniform point, while generating the punctured key at the new pair (x^*, y^*) . Formally, we define the following probability,

$$p_2 := \Pr \left[\mathbf{A}(\text{pk}^*, \text{td}^*\{y^*\}, y^*) = x^* \mid \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ x^* \leftarrow [N] \\ k\{x^*, y^*\} \leftarrow \text{punc.Punc}(k, x^*, y^*) \\ \text{pk}^* \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{eval}}[k\{x^*, y^*\}, x^*, y^*]) \\ \text{td}^*\{y^*\} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{inv}}[k\{x^*, y^*\}, y^*]) \end{array} \right].$$

By the puncturable PRP security,

$$|p_2 - p_1| \leq \epsilon_{\text{punc}}(\lambda, N).$$

Hybrid 3: Restoring the base key. Replace both circuits by the following circuits, which use the base key k :

$C_{\text{eval}}[k, x^*, y^*](x)$:

1. If $x = x^*$, output y^* .
2. Otherwise, output `base. $\Pi(k, x)$` .

$C_{\text{inv}}[k, y^*](y)$:

1. If $y = y^*$, output $\perp$.
2. Otherwise, output `base. $\Pi^{-1}(k, y)$` .

The success probability in this hybrid is

$$p_3 := \Pr \left[\mathbf{A}(\text{pk}^*, \text{td}^*\{y^*\}, y^*) = x^* \mid \begin{array}{l} k \leftarrow \{0, 1\}^\lambda \\ x^* \leftarrow [N] \\ \text{pk}^* \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{eval}}[k, x^*, y^*]) \\ \text{td}^*\{y^*\} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{inv}}[k, y^*]) \end{array} \right].$$

Puncturing correctness preserves forward evaluation at every input other than x^* and inverse evaluation at every input other than y^* . At the punctured points, both pairs of circuits use the same explicit branches. Thus, the corresponding circuits are functionally equivalent, and `iO` security gives

$$|p_3 - p_2| \leq 2\epsilon_{\text{iO}}(\lambda).$$

Final step: Unpredictability of the designated point. In Hybrid 3, x^* is uniform and independent of k and the fixed output y^* . Set

$$C(x) := \text{base}.\Pi(k, x), \quad C'(x) := y^*, \quad R(x) := x.$$

The forward circuit is exactly the joint program

$$J_{R=x^*}[C, C'] \equiv C_{\text{eval}}[k, x^*, y^*].$$

Apply the unpredictability guarantee of Lemma A.1 with one circuit and uniform trigger x^* . The injective length-doubling PRG witnesses hardness of the uniform distribution with error $\epsilon_{\text{PRG}}(N)$. Given the description of C and the joint-program obfuscation, the predictor proceeds as follows:

1. Read k from the description of C and set pk^* to the supplied obfuscation.
2. Generate $td^*\{y^*\} \leftarrow iO(1^\lambda, 1^s, C_{inv}[k, y^*])$.
3. Run $\mathbf{A}(pk^*, td^*\{y^*\}, y^*)$ and output its answer.

For each fixed k , this gives exactly the Hybrid 3 view, and the predictor succeeds exactly when the event defining p_3 occurs. Averaging the lemma's bound over k gives

$$p_3 \leq \frac{1}{N} + \epsilon_{PRG}(N) + \epsilon_{iO}(\lambda).$$

Combining the transitions,

$$p_0 \leq \frac{1}{N} + \epsilon_{PRG}(N) + 5\epsilon_{iO}(\lambda) + \epsilon_{punc}(\lambda, N).$$

By Theorem 4.2,

$$\epsilon_{punc}(\lambda, N) \leq O\left(\frac{1}{N} + \epsilon_{PRG}(N) + \epsilon_{iO}(\lambda) + \epsilon_{PPRP}(\lambda, N)\right).$$

Consequently,

$$p_0 \leq O\left(\frac{1}{N} + \epsilon_{PRG}(N) + \epsilon_{iO}(\lambda) + \epsilon_{PPRP}(\lambda, N)\right) = \text{negl}(\lambda),$$

since N is exponential in λ and the underlying primitives are secure. This proves puncturable trapdoor security.

Puncturable trapdoor hiding. Fix a PPT adversary $\mathbf{A}$. The proof is a direct implication of the sparse trigger lemma (Lemma A.1). For every $k \in \{0, 1\}^\lambda$, define the base circuit C_k , alternative circuit C' , and trigger function R :

$$C_k(y) := \text{base}.\Pi^{-1}(k, y) \ , \quad C'(\cdot) := \perp \ , \quad R(y) := y \ .$$

For every $y^* \in [N]$, the corresponding joint program is precisely the punctured inverse circuit:

$$J_{R=y^*}[C_k, C'] \equiv C_{inv}[k][y^* \rightarrow \perp] \ .$$

Suppose that $\mathbf{A}$, given the public key pk , the original trapdoor td , and the punctured trapdoor $td\{y^*\}$, predicts y^* with some probability. We construct an adversary $\mathbf{A}'$ to the unpredictability of Lemma A.1 with $\ell = 1$. Given the description of C_k and the obfuscation $\widehat{J}$ of $J_{R=y^*}[C_k, C']$, the predictor $\mathbf{A}'$ does the following:

1. Parse the hard-coded value k from the description of C_k .
2. Compute $\widehat{C_{eval}} \leftarrow iO(1^\lambda, 1^s, C_{eval}[k])$.
3. Set $pk := \widehat{C_{eval}}$, $td := k$, and $td\{y^*\} := \widehat{J}$.
4. Emulate and output $y' \leftarrow \mathbf{A}(pk, td, td\{y^*\})$.

The view emulated by $\mathbf{A}'$ is distributed exactly as $(pk, td, td\{y^*\})$ in the hiding experiment. Therefore, the success probabilities of $\mathbf{A}$ and $\mathbf{A}'$ are equal. For every fixed k , the unpredictability guarantee

of Lemma A.1 bounds this probability. Since the bound is independent of k , it continues to hold when $k \leftarrow \{0, 1\}^\lambda$ is sampled as part of **Gen**. Hence,

$$\Pr \left[y' = y^* \mid \begin{array}{l} (\text{pk}, \text{td}) \leftarrow \text{Gen}(1^\lambda, N) \\ y^* \leftarrow [N] \\ \text{td}\{y^*\} \leftarrow \text{Punc}(\text{td}, y^*) \\ y' \leftarrow \mathbf{A}(\text{pk}, \text{td}, \text{td}\{y^*\}) \end{array} \right] \leq \frac{1}{N} + \epsilon_{\text{PRG}}(N) + \epsilon_{\text{iO}}(\lambda) .$$

Since N is exponential in λ , by the security of the underlying PRG, iO, and PPRP,

$$\frac{1}{N} = \text{negl}(\lambda), \quad \epsilon_{\text{PRG}}(N) = \text{negl}(\lambda), \quad \epsilon_{\text{iO}}(\lambda) = \text{negl}(\lambda), \quad \epsilon_{\text{PPRP}}(\lambda, N) = \text{negl}(\lambda).$$

Substituting these security errors into the bounds above gives

$$\epsilon(\lambda, N) \leq \text{negl}(\lambda) .$$

□

5.4 Proof of Theorem 5.5

Proof. Fix an $(\epsilon_{\mathcal{D}}, q)$ -hard source ensemble $\mathcal{D}$ and a non-uniform PPT distinguisher $\mathbf{A}$. Let δ_{real} and δ_{rand} be its acceptance probabilities in the experiments of Definition 5.3. Apply Theorem 4.2 to the underlying PPRP **base** and write **punc** for the resulting puncturable PRP. Its base forward and inverse algorithms are those of **base**. Let $\epsilon_{\text{iO}}(\lambda)$, $\epsilon_{\text{punc}}(\lambda, N)$, $\epsilon_{\text{PRG}}(N)$, and $\epsilon_{\text{PPRP}}(\lambda, N)$ be the security parameters of the assumed indistinguishability obfuscation, puncturable PRP, PRG, and permutable PRP, respectively.

The proof follows the same sequence as the proof of Theorem 5.4: puncture the key, randomize the designated value, and restore the key. Here, we replace the image y^* of the source point $x^* \leftarrow \mathcal{D}$ by an independent uniform output. After restoring the key, the forward circuit retains a special branch that maps x^* to y^* . Finally, the indistinguishability guarantee of Lemma A.1 lets us remove this branch, restoring the original evaluation circuit while keeping the challenge output uniform.

Hybrid 0: The real experiment. In this hybrid, we use the following base circuits:

$C_{\text{eval}}[k](x)$:

1. Output **base**. $\Pi(k, x)$.

$C_{\text{inv}}[k, y^*](y)$:

1. If $y = y^*$, output $\perp$.
2. Otherwise, output **base**. $\Pi^{-1}(k, y)$.

The acceptance probability in this hybrid is

$$\delta_0 := \Pr \left[\mathbf{A}(\text{pk}^*, \text{td}^*\{y^*\}, y^*) = 1 \mid \begin{array}{l} x^* \leftarrow \mathcal{D}_N \\ k \leftarrow \{0, 1\}^\lambda \\ y^* \leftarrow \text{base}.\Pi(k, x^*) \\ \text{pk}^* \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{eval}}[k]) \\ \text{td}^*\{y^*\} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{inv}}[k, y^*]) \end{array} \right] .$$

Thus $\delta_0 = \delta_{\text{real}}$.

Hybrid 1: Puncturing the base key. Use the following circuits:

$C_{\text{eval}}[\mathbf{k}\{x^*, y^*\}, x^*, y^*](x)$:

1. If $x = x^*$, output y^* .
2. Otherwise, output $\text{punc.Eval}(\mathbf{k}\{x^*, y^*\}, x)$.

$C_{\text{inv}}[\mathbf{k}\{x^*, y^*\}, y^*](y)$:

1. If $y = y^*$, output $\perp$.
2. Otherwise, output $\text{punc.Eval}^{-1}(\mathbf{k}\{x^*, y^*\}, y)$.

The experiment is

$$\delta_1 := \Pr \left[\mathbf{A}(\text{pk}^*, \text{td}^*\{y^*\}, y^*) = 1 \mid \begin{array}{l} x^* \leftarrow \mathcal{D}_N \\ \mathbf{k} \leftarrow \{0, 1\}^\lambda \\ y^* \leftarrow \text{base.II}(\mathbf{k}, x^*) \\ \mathbf{k}\{x^*, y^*\} \leftarrow \text{punc.Punc}(\mathbf{k}, x^*, y^*) \\ \text{pk}^* \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{eval}}[\mathbf{k}\{x^*, y^*\}, x^*, y^*]) \\ \text{td}^*\{y^*\} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{inv}}[\mathbf{k}\{x^*, y^*\}, y^*]) \end{array} \right].$$

Puncturing correctness makes both circuits functionally equivalent to those of Hybrid 0, including at their explicitly handled points. Hence

$$|\delta_1 - \delta_0| \leq 2\epsilon_{\text{iO}}(\lambda).$$

Hybrid 2: Randomizing the challenge image. Keep these circuits, but sample y^* independently:

$$\delta_2 := \Pr \left[\mathbf{A}(\text{pk}^*, \text{td}^*\{y^*\}, y^*) = 1 \mid \begin{array}{l} x^* \leftarrow \mathcal{D}_N \\ \mathbf{k} \leftarrow \{0, 1\}^\lambda \\ y^* \leftarrow [N] \\ \mathbf{k}\{x^*, y^*\} \leftarrow \text{punc.Punc}(\mathbf{k}, x^*, y^*) \\ \text{pk}^* \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{eval}}[\mathbf{k}\{x^*, y^*\}, x^*, y^*]) \\ \text{td}^*\{y^*\} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{inv}}[\mathbf{k}\{x^*, y^*\}, y^*]) \end{array} \right].$$

Given $(x^*, \mathbf{k}\{x^*, y^*\}, y^*)$, the reduction can generate both obfuscations and supply the entire view of $\mathbf{A}$. Input-puncturing security of punc therefore gives

$$|\delta_2 - \delta_1| \leq \epsilon_{\text{punc}}(\lambda, N).$$

As above, this reduction fixes x^* as non-uniform advice and averages the resulting bound. It does not need to sample the source.

Hybrid 3: Restoring the base key. Replace both circuits by the following circuits, which use the base key $\mathbf{k}$:

$C_{\text{eval}}[\mathbf{k}, x^*, y^*](x)$:

1. If $x = x^*$, output y^* .
2. Otherwise, output $\text{base.II}(\mathbf{k}, x)$.

$C_{\text{inv}}[\mathbf{k}, y^*](y)$:

1. If $y = y^*$, output $\perp$.
2. Otherwise, output $\text{base}.\Pi^{-1}(\mathbf{k}, y)$.

The acceptance probability in this hybrid is

$$\delta_3 := \Pr \left[\mathbf{A}(\text{pk}^*, \text{td}^*\{y^*\}, y^*) = 1 \left| \begin{array}{l} x^* \leftarrow \mathcal{D}_N \\ \mathbf{k} \leftarrow \{0, 1\}^\lambda \\ y^* \leftarrow [N] \\ \text{pk}^* \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{eval}}[\mathbf{k}, x^*, y^*]) \\ \text{td}^*\{y^*\} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{inv}}[\mathbf{k}, y^*]) \end{array} \right. \right].$$

Puncturing correctness preserves forward evaluation at every input other than x^* and inverse evaluation at every input other than y^* . At the punctured points, both pairs of circuits use the same explicit branches. Thus, the corresponding circuits are functionally equivalent, and iO security gives

$$|\delta_3 - \delta_2| \leq 2\epsilon_{\text{iO}}(\lambda).$$

Hybrid 4: Removing the source-dependent branch. Remove the branch for x^* from the selected evaluation circuit, restoring the circuit of Hybrid 0. The acceptance probability in this hybrid is

$$\delta_4 := \Pr \left[\mathbf{A}(\text{pk}^*, \text{td}^*\{y^*\}, y^*) = 1 \left| \begin{array}{l} \mathbf{k} \leftarrow \{0, 1\}^\lambda \\ y^* \leftarrow [N] \\ \text{pk}^* \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{eval}}[\mathbf{k}]) \\ \text{td}^*\{y^*\} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{inv}}[\mathbf{k}, y^*]) \end{array} \right. \right].$$

Apply Lemma A.1 with trigger $R(x) = x$, base circuit $C_{\text{eval}}[\mathbf{k}]$, alternative circuit $C'(x) = y^*$, and trigger value $x^* \leftarrow \mathcal{D}_N$. The key $\mathbf{k}$, the uniform challenge y^* , and the randomness used to obfuscate the punctured-inverse circuit are independent of the trigger point $x^* \leftarrow \mathcal{D}_N$. Fixing these values as non-uniform advice lets the lemma's distinguisher supply $\text{td}^*\{y^*\}$ and y^* alongside its challenge obfuscation. Averaging over y^* gives

$$|\delta_4 - \delta_3| \leq \frac{1}{N} + \epsilon_{\mathcal{D}}(N) + 2\epsilon_{\text{iO}}(\lambda).$$

No efficient sampler for $\mathcal{D}$ is required. At this point, the selected public key and punctured trapdoor are honestly generated, and y^* is uniform and independent of the selected base key.

Thus $\delta_4 = \delta_{\text{rand}}$. Combining the transitions,

$$|\delta_{\text{real}} - \delta_{\text{rand}}| \leq \frac{1}{N} + \epsilon_{\mathcal{D}}(N) + 6\epsilon_{\text{iO}}(\lambda) + \epsilon_{\text{punc}}(\lambda, N).$$

By Theorem 4.2,

$$\epsilon_{\text{punc}}(\lambda, N) \leq O\left(\frac{1}{N} + \epsilon_{\text{PRG}}(N) + \epsilon_{\text{iO}}(\lambda) + \epsilon_{\text{PPRP}}(\lambda, N)\right).$$

Since N is exponential in λ , security of the underlying primitives gives

$$|\delta_{\text{real}} - \delta_{\text{rand}}| \leq O(\epsilon_{\mathcal{D}}(N)) + \text{negl}(\lambda),$$

as required. □

5.5 Proof of Corollary 5.6

Proof. Fix an $(\epsilon_{\mathcal{D}}, q)$ -hard source ensemble $\mathcal{D} = \{\mathcal{D}_N\}_{N \in \mathbb{N}}$ independent of key generation, an efficiently samplable domain-size function $N = N(\lambda)$, and a PPT inverter $\mathbf{A}$ in Definition 5.2. Let p_{src} be its success probability:

$$p_{\text{src}} := \Pr \left[\hat{x} = x^* \left| \begin{array}{l} x^* \leftarrow \mathcal{D}_N \\ (\text{pk}, \text{td}) \leftarrow \text{Gen}(1^\lambda, N) \\ y^* \leftarrow \Pi(\text{pk}, x^*) \\ \text{td}\{y^*\} \leftarrow \text{Punc}(\text{td}, y^*) \\ \hat{x} \leftarrow \mathbf{A}(\text{pk}, \text{td}\{y^*\}, y^*) \end{array} \right. \right].$$

Construct a distinguisher $\mathbf{A}'$ for Definition 5.3. On input $(\text{pk}, \text{td}\{y^*\}, y^*)$, it proceeds as follows:

1. Run $\hat{x} \leftarrow \mathbf{A}(\text{pk}, \text{td}\{y^*\}, y^*)$.
2. If $\Pi(\text{pk}, \hat{x}) = y^*$, then output 1.
3. Else, output 0.

Let δ_{real} and δ_{rand} be the acceptance probabilities in the two experiments of Definition 5.3. Explicitly,

$$\delta_{\text{real}} := \Pr \left[\mathbf{A}'(\text{pk}, \text{td}\{y^*\}, y^*) = 1 \left| \begin{array}{l} x^* \leftarrow \mathcal{D}_N \\ (\text{pk}, \text{td}) \leftarrow \text{Gen}(1^\lambda, N) \\ y^* \leftarrow \Pi(\text{pk}, x^*) \\ \text{td}\{y^*\} \leftarrow \text{Punc}(\text{td}, y^*) \end{array} \right. \right],$$

$$\delta_{\text{rand}} := \Pr \left[\mathbf{A}'(\text{pk}, \text{td}\{y^*\}, y^*) = 1 \left| \begin{array}{l} (\text{pk}, \text{td}) \leftarrow \text{Gen}(1^\lambda, N) \\ y^* \leftarrow [N] \\ \text{td}\{y^*\} \leftarrow \text{Punc}(\text{td}, y^*) \end{array} \right. \right].$$

In the real experiment, $y^* = \Pi(\text{pk}, x^*)$. For every honestly generated public key, evaluation is a permutation of $[N]$. Therefore, for every valid $\hat{x}$,

$$\Pi(\text{pk}, \hat{x}) = y^* \iff \hat{x} = x^*.$$

We get that,

$$\begin{aligned} \delta_{\text{real}} &= \Pr \left[\Pi(\text{pk}, \hat{x}) = y^* \left| \begin{array}{l} x^* \leftarrow \mathcal{D}_N \\ (\text{pk}, \text{td}) \leftarrow \text{Gen}(1^\lambda, N) \\ y^* \leftarrow \Pi(\text{pk}, x^*) \\ \text{td}\{y^*\} \leftarrow \text{Punc}(\text{td}, y^*) \\ \hat{x} \leftarrow \mathbf{A}(\text{pk}, \text{td}\{y^*\}, y^*) \end{array} \right. \right] \\ &= \Pr \left[\hat{x} = x^* \left| \begin{array}{l} x^* \leftarrow \mathcal{D}_N \\ (\text{pk}, \text{td}) \leftarrow \text{Gen}(1^\lambda, N) \\ y^* \leftarrow \Pi(\text{pk}, x^*) \\ \text{td}\{y^*\} \leftarrow \text{Punc}(\text{td}, y^*) \\ \hat{x} \leftarrow \mathbf{A}(\text{pk}, \text{td}\{y^*\}, y^*) \end{array} \right. \right] \\ &= p_{\text{src}}. \end{aligned}$$

The first equality follows from the definition of $\mathbf{A}'$. The second follows because $\Pi(\mathbf{pk}, \cdot)$ is a permutation and $y^* = \Pi(\mathbf{pk}, x^*)$. The final equality is the definition of p_{src} . By weak-source pseudorandomness,

$$p_{\text{src}} = \delta_{\text{real}} \leq \delta_{\text{rand}} + \epsilon_{\text{src-pr}}(\lambda, N, \epsilon_{\mathcal{D}}).$$

To bound δ_{rand} , fix any $y \in [N]$ independently of key generation and define

$$p_y := \Pr \left[\Pi(\mathbf{pk}, \hat{x}) = y \mid \begin{array}{l} (\mathbf{pk}, \text{td}) \leftarrow \text{Gen}(1^\lambda, N) \\ \text{td}\{y\} \leftarrow \text{Punc}(\text{td}, y) \\ \hat{x} \leftarrow \mathbf{A}(\mathbf{pk}, \text{td}\{y\}, y) \end{array} \right].$$

This is precisely the success probability of $\mathbf{A}$ in the puncturable trapdoor security experiment of Definition 5.1. By the assumed puncturable trapdoor security, $p_y \leq \epsilon(\lambda, N)$ for every y . In the random experiment, y^* is uniform and independent of key generation. Conditional on $y^* = y$, its remaining distribution is exactly the experiment defining p_y . Consequently,

$$\delta_{\text{rand}} = \frac{1}{N} \sum_{y \in [N]} p_y \leq \epsilon(\lambda, N).$$

Combining the two bounds gives

$$p_{\text{src}} \leq \epsilon_{\text{src-pr}}(\lambda, N, \epsilon_{\mathcal{D}}) + \epsilon(\lambda, N).$$

This proves the general implication. For the stated instantiation, Theorems 5.4 and 5.5 give the bound $O(\epsilon_{\mathcal{D}}(N)) + \text{negl}(\lambda)$. $\square$

5.6 Application - PPAD Hardness

The class PPAD [Pap94] consists of all total search problems that are polynomial-time reducible to the END-OF-THE-LINE problem (EOL).

Definition 5.8 (END-OF-THE-LINE [BPR15]). *An EOL instance is a tuple $(y^*, \mathbf{S}, \mathbf{P})$, where $y^* \in \{0, 1\}^\lambda$ and $\mathbf{S}, \mathbf{P} : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda$ are circuits. A valid instance satisfies $\mathbf{P}(y^*) = y^* \neq \mathbf{S}(y^*)$. Moreover, a string $w \in \{0, 1\}^\lambda$ is a valid witness if*

$$\mathbf{P}(\mathbf{S}(w)) \neq w \quad \vee \quad (w \neq y^* \wedge \mathbf{S}(\mathbf{P}(w)) \neq w) . \quad (1)$$

For an invalid instance that does not satisfy the condition $\mathbf{P}(y^*) = y^* \neq \mathbf{S}(y^*)$, we say that y^* itself is a valid witness.

Construction 5.9 (EOL distribution from TDPs with puncturable trapdoors). Let $(\text{Gen}, \Pi, \Pi^{-1}, \text{Punc})$ be an ϵ -secure TDP with a puncturable trapdoor as in Definition 5.1. For every security parameter $\lambda \in \mathbb{N}$, set $N := 2^\lambda$ and fix $y^* := 1$. For $(\mathbf{pk}, \text{td}) \leftarrow \text{Gen}(1^\lambda, N)$, let $\text{td}\{y^*\} \leftarrow \text{Punc}(\text{td}, y^*)$, and define the following circuits:

- $\mathbf{S}_{\mathbf{pk}}(w)$: Output $\Pi(\mathbf{pk}, w)$.
- $\mathbf{P}_{\text{td}\{y^*\}, y^*}(y)$:

1. If $y = y^*$, then output y^* .
2. Else, output $\Pi^{-1}(\text{td}\{y^*\}, y)$.

We define the distribution $\mathcal{D}_{\text{EOL}}$ by

$$\mathcal{D}_{\text{EOL}}(1^\lambda) := \left\{ (y^*, S, P) \left| \begin{array}{l} y^* := 1 \\ (\text{pk}, \text{td}) \leftarrow \text{Gen}(1^\lambda, N) \\ \text{td}\{y^*\} \leftarrow \text{Punc}(\text{td}, y^*) \\ S := S_{\text{pk}} \\ P := P_{\text{td}\{y^*\}, y^*} \end{array} \right. \right\} .$$

Note that the choice $y^* = 1$ is arbitrary. Any fixed point in $[N]$ that is chosen independently of (pk, td) gives the same security guarantee.

Theorem 5.10. *Assume the existence of an ϵ -secure TDP with a puncturable trapdoor as in Definition 5.1. For every security parameter $\lambda \in \mathbb{N}$, set $N := 2^\lambda$. Then, $\mathcal{D}_{\text{EOL}}(1^\lambda)$ from Construction 5.9 is an efficiently samplable distribution over EOL instances that are valid except with probability at most $\epsilon(\lambda, N)$. Witness validity includes the convention that y^* is a valid witness for an invalid instance. For every PPT adversary $\mathbf{A}$,*

$$\Pr \left[w \text{ is a valid EOL witness} \mid \begin{array}{l} (y^*, S, P) \leftarrow \mathcal{D}_{\text{EOL}}(1^\lambda) \\ w \leftarrow \mathbf{A}(y^*, S, P) \end{array} \right] \leq \epsilon(\lambda, N) .$$

Therefore, if $\epsilon(\lambda, N)$ is negligible in λ , then $\mathcal{D}_{\text{EOL}}$ is a hard distribution over EOL instances, and hence gives a hard-on-average problem in PPA Δ under the convention above.

Corollary 5.11. *Assume the existence of an indistinguishability obfuscation scheme, an injective length-doubling pseudorandom generator, and a PPRP for the class of general swaps. Then, the EOL problem is hard on average.*

Proof of Corollary 5.11. Let $\lambda \in \mathbb{N}$ be the security parameter. Let $\epsilon_{\text{IO}}(\lambda)$, $\epsilon_{\text{PRG}}(N)$, and $\epsilon_{\text{PRP}}(\lambda, N)$ denote the security errors of the indistinguishability obfuscation scheme, the pseudorandom generator, and the PPRP, respectively. Set $N := 2^\lambda$. By Theorem 5.4, these primitives yield an ϵ -secure TDP over $[N]$ with a puncturable trapdoor, where

$$\begin{aligned} \epsilon(\lambda, N) &\leq O\left(\frac{1}{N} + \epsilon_{\text{PRG}}(N) + \epsilon_{\text{IO}}(\lambda) + \epsilon_{\text{PRP}}(\lambda, N)\right) \\ &= \text{negl}(\lambda) . \end{aligned}$$

The final equality follows from the security of the three underlying primitives and the fact that $1/N = 2^{-\lambda}$ is negligible. Hence, Theorem 5.10 implies that $\mathcal{D}_{\text{EOL}}$ is an efficiently samplable hard distribution over EOL instances, valid except with negligible probability, and that, for every PPT adversary $\mathbf{A}$,

$$\Pr \left[w \text{ is a valid EOL witness} \mid \begin{array}{l} (y^*, S, P) \leftarrow \mathcal{D}_{\text{EOL}}(1^\lambda) \\ w \leftarrow \mathbf{A}(y^*, S, P) \end{array} \right] \leq \epsilon(\lambda, N) = \text{negl}(\lambda) .$$

Therefore, the EOL problem is hard on average. $\square$

Proof of Theorem 5.10. By construction, $P_{\text{td}\{y^*\}, y^*}(y^*) = y^*$. Thus, the sampled instance is invalid precisely when $\Pi(\text{pk}, y^*) = y^*$. An inverter that always outputs y^* succeeds on this event, so puncturable trapdoor security bounds its probability by $\epsilon(\lambda, N)$. On this event, our convention makes y^* a valid witness, and it is also a preimage of itself. Moreover, the distribution is efficiently samplable because **Gen** and **Punc** are PPT algorithms and the two circuits in Construction 5.9 have polynomial size. We now move to proving hardness.

Hardness. Let $N := 2^\lambda$, and let $\delta(\lambda)$ be the success probability of a PPT adversary **A** in the experiment from the statement of the theorem. We construct an adversary **A'** against puncturable trapdoor security. Given $(\text{pk}, \text{td}\{y^*\}, y^*)$, the adversary **A'** constructs S_{pk} and $P_{\text{td}\{y^*\}, y^*}$, emulates

$$w \leftarrow \mathbf{A}(y^*, S_{\text{pk}}, P_{\text{td}\{y^*\}, y^*}) ,$$

and outputs w . The instance given to **A** is distributed exactly according to $\mathcal{D}_{\text{EOL}}(1^\lambda)$.

We next analyze the output of **A** and show that every valid EOL witness is a preimage of y^* under $\Pi(\text{pk}, \cdot)$. Fix keys (pk, td) and the corresponding punctured trapdoor $\text{td}\{y^*\}$, and fix $w \in [N]$. Since $S_{\text{pk}}(\cdot) \equiv \Pi(\text{pk}, \cdot)$ is a permutation, and by the correctness of the puncturable trapdoor,

$$\forall x \in [N] \text{ s.t. } S_{\text{pk}}(x) \neq y^*, \quad P_{\text{td}\{y^*\}, y^*}(S_{\text{pk}}(x)) = x . \quad (2)$$

$$\forall y \in [N] \setminus \{y^*\}, \quad S_{\text{pk}}(P_{\text{td}\{y^*\}, y^*}(y)) = y . \quad (3)$$

We now prove that w is a valid EOL witness if and only if $S_{\text{pk}}(w) = y^*$. We split first according to whether w is a preimage of y^* , and then according to whether $w = y^*$.

- $S_{\text{pk}}(w) = y^*$: In this case, w is a preimage of y^* . We consider two subcases.
 - If $w = y^*$, then the instance is invalid because $S_{\text{pk}}(y^*) = y^*$. Hence, w is a valid EOL witness by our convention.
 - If $w \neq y^*$, then

$$\begin{aligned} P_{\text{td}\{y^*\}, y^*}(S_{\text{pk}}(w)) &= P_{\text{td}\{y^*\}, y^*}(y^*) \\ &= y^* \neq w . \end{aligned}$$

Thus, the first condition in (1) holds, and w is a valid EOL witness.

Therefore, whenever $S_{\text{pk}}(w) = y^*$, w is a valid EOL witness.

- $S_{\text{pk}}(w) \neq y^*$: In this case, (2) gives

$$P_{\text{td}\{y^*\}, y^*}(S_{\text{pk}}(w)) = w .$$

Hence, the first condition in (1) does not hold. We again consider two subcases.

- If $w = y^*$, then the instance is valid because $S_{\text{pk}}(y^*) \neq y^*$, and the second condition does not hold because $w = y^*$. Thus, w is not a valid EOL witness.
- If $w \neq y^*$, then the first condition does not hold. Moreover, by (3),

$$S_{\text{pk}}(P_{\text{td}\{y^*\}, y^*}(w)) = w ,$$

so the second condition does not hold. The invalid-instance convention cannot apply to $w \neq y^*$. Thus, w is not a valid EOL witness.

Therefore, whenever $S_{\mathbf{pk}}(w) \neq y^*$, w is not a valid EOL witness.

The first case proves that

$$S_{\mathbf{pk}}(w) = y^* \implies w \text{ is a valid EOL witness .}$$

The second case proves the contrapositive of the reverse implication, and therefore

$$w \text{ is a valid EOL witness } \implies S_{\mathbf{pk}}(w) = y^* .$$

Combining the two implications, and recalling that $S_{\mathbf{pk}}(w) = \Pi(\mathbf{pk}, w)$ by construction, we obtain

$$w \text{ is a valid EOL witness } \iff S_{\mathbf{pk}}(w) = y^* \iff \Pi(\mathbf{pk}, w) = y^* . \quad (4)$$

In particular, since $S_{\mathbf{pk}}$ is a permutation, the sampled instance has the unique witness

$$w^* := \Pi^{-1}(\mathbf{td}, y^*) .$$

By (4), whenever $\mathbf{A}$ outputs a valid EOL witness, $\mathbf{A}'$ outputs a preimage of y^* . Therefore, puncturable trapdoor security for the fixed point $y^* = 1$ gives

$$\delta(\lambda) \leq \epsilon(\lambda, N) ,$$

as required. Notice that this application uses only puncturable trapdoor security and correctness (puncturable trapdoor hiding is not needed). $\square$

6 TDP with a full key domain in the CRS model

In this section, we lift the TDP with a puncturable trapdoor from Section 5 to obtain full key and message domains in the CRS model. The central requirement is that, under every honestly generated crs , every string in the key space defines a permutation of the full message domain.

The master key derives a trapdoor for each key and can be punctured to restrict derivation at one selected key. Each individual trapdoor can also be punctured at a selected output. We require two-level punctured security, which means that inversion remains hard for a uniformly random key and challenge output even given both the master key punctured at that key and the corresponding trapdoor punctured at that output. In addition to two-level punctured security for a uniformly random challenge, we define and prove two-level punctured security when the challenge is sampled from a weak source, together with a pseudorandom variant of this weak-source notion.

6.1 Definition

Definition 6.1 (Trapdoor permutation with a full key domain in the CRS model). *Let $\epsilon : \mathbb{N} \times \mathbb{N} \rightarrow [0, 1]$ be a security-error function. An ϵ -secure trapdoor one-way permutation in the CRS model is a tuple of algorithms*

$$(\text{Gen}, \Pi, \Pi^{-1}, \text{Trapdoor}, \text{PuncMK}, \text{PuncTD})$$

with the following properties:

- **Setup.** *The Gen algorithm is a PPT algorithm that takes as input a security parameter 1^λ and a domain size N , where $\lambda, N \in \mathbb{N}$, and outputs a common reference string crs and a master key mk .*
- **Efficient permutations.** *The algorithm Π is a deterministic polynomial-time algorithm that given a common reference string crs and a key $\mathbf{k} \leftarrow \{0, 1\}^{\ell(\lambda, \log N)}$ computes a permutation over $[N]$.*
- **Full key-domain correctness.** *For every $\lambda, N \in \mathbb{N}$, $(\text{crs}, \cdot) \in \text{Gen}(1^\lambda, N)$, and for every string $\mathbf{k} \in \{0, 1\}^{\ell(\lambda, \log N)}$, the function $\Pi(\text{crs}, \mathbf{k}, \cdot)$ is a permutation over $[N]$.*
- **Trapdoor.** *The algorithm Trapdoor is deterministic.*

For every $\lambda, N \in \mathbb{N}$, for every $\mathbf{k} \in \{0, 1\}^{\ell(\lambda, \log N)}$, and for every $x \in [N]$,

$$\Pr \left[\Pi^{-1}(\text{td}, \Pi(\text{crs}, \mathbf{k}, x)) = x \mid \begin{array}{l} (\text{crs}, \text{mk}) \leftarrow \text{Gen}(1^\lambda, N) \\ \text{td} \leftarrow \text{Trapdoor}(\text{mk}, \mathbf{k}) \end{array} \right] = 1 .$$

- **Punctured key correctness and consistency.** *For every $\lambda, N \in \mathbb{N}$, every $x \in [N]$, and every $\mathbf{k}, \mathbf{k}' \in \{0, 1\}^{\ell(\lambda, \log N)}$ such that $\mathbf{k} \neq \mathbf{k}'$,*

$$\Pr \left[\begin{array}{l} \text{td} = \text{Trapdoor}(\text{mk}, \mathbf{k}) \\ \wedge \Pi^{-1}(\text{td}, \Pi(\text{crs}, \mathbf{k}, x)) = x \end{array} \mid \begin{array}{l} (\text{crs}, \text{mk}) \leftarrow \text{Gen}(1^\lambda, N) \\ \text{mk}\{\mathbf{k}'\} \leftarrow \text{PuncMK}(\text{mk}, \mathbf{k}') \\ \text{td} \leftarrow \text{Trapdoor}(\text{mk}\{\mathbf{k}'\}, \mathbf{k}) \end{array} \right] = 1 .$$

- **Punctured trapdoor correctness.** *The PPT algorithm $\text{PuncTD}(\text{td}, y')$ punctures the trapdoor td at y' . For every $\lambda, N \in \mathbb{N}$, every $\mathbf{k} \in \{0, 1\}^{\ell(\lambda, \log N)}$, and every $y, y' \in [N]$ such that $y \neq y'$,*

$$\Pr \left[\begin{array}{l} \Pi(\text{crs}, \mathbf{k}, \Pi^{-1}(\text{td}\{y'\}, y)) = y \\ \wedge \Pi^{-1}(\text{td}\{y'\}, y') = \perp \end{array} \mid \begin{array}{l} (\text{crs}, \text{mk}) \leftarrow \text{Gen}(1^\lambda, N) \\ \text{td} \leftarrow \text{Trapdoor}(\text{mk}, \mathbf{k}) \\ \text{td}\{y'\} \leftarrow \text{PuncTD}(\text{td}, y') \end{array} \right] = 1 .$$

- Two-level punctured security. For every PPT adversary $\mathbf{A}$ and every efficiently computable domain-size function $N = N(\lambda)$ that is exponential in λ , for every security parameter λ and every fixed $y \in [N]$,

$$\Pr \left[\Pi(\text{crs}, k, x) = y \mid \begin{array}{l} (\text{crs}, \text{mk}) \leftarrow \text{Gen}(1^\lambda, N) \\ k \leftarrow \{0, 1\}^{\ell(\lambda, \log N)} \\ \text{mk}\{k\} \leftarrow \text{PuncMK}(\text{mk}, k) \\ \text{td} \leftarrow \text{Trapdoor}(\text{mk}, k) \\ \text{td}\{y\} \leftarrow \text{PuncTD}(\text{td}, y) \\ x \leftarrow \mathbf{A}(\text{crs}, y, \text{mk}\{k\}, k, \text{td}\{y\}) \end{array} \right] \leq \epsilon(\lambda, N) .$$

We also write $\text{PuncTD}(\text{mk}, k, y)$ as shorthand for $\text{PuncTD}(\text{Trapdoor}(\text{mk}, k), y)$.

Definition 6.2 (Weak-source two-level punctured security in the CRS model). A dense-key trapdoor permutation in the CRS model is said to satisfy weak-source two-level punctured security with error ϵ_{src} if for every ensemble of distributions $\mathcal{D} := \{\mathcal{D}_N\}_{N \in \mathbb{N}}$ that is $(\epsilon_{\mathcal{D}}, q)$ -hard, for every PPT adversary $\mathbf{A}$, every security parameter $\lambda \in \mathbb{N}$, and every domain size $N \in \mathbb{N}$,

$$\Pr \left[\hat{x} = x \mid \begin{array}{l} x \leftarrow \mathcal{D}_N \\ (\text{crs}, \text{mk}) \leftarrow \text{Gen}(1^\lambda, N) \\ k \leftarrow \{0, 1\}^{\ell(\lambda, \log N)} \\ y \leftarrow \Pi(\text{crs}, k, x) \\ \text{mk}\{k\} \leftarrow \text{PuncMK}(\text{mk}, k) \\ \text{td} \leftarrow \text{Trapdoor}(\text{mk}, k) \\ \text{td}\{y\} \leftarrow \text{PuncTD}(\text{td}, y) \\ \hat{x} \leftarrow \mathbf{A}(\text{crs}, y, \text{mk}\{k\}, k, \text{td}\{y\}) \end{array} \right] \leq \epsilon_{\text{src}}(\lambda, N, \epsilon_{\mathcal{D}}) .$$

Definition 6.3 (Weak-source pseudorandomness in the CRS model). A dense-key trapdoor permutation in the CRS model is $\epsilon_{\text{src-pr}}$ -pseudorandom if for every ensemble of distributions $\mathcal{D} := \{\mathcal{D}_N\}_{N \in \mathbb{N}}$ that is $(\epsilon_{\mathcal{D}}, q)$ -hard, for every PPT adversary $\mathbf{A}$, every security parameter $\lambda \in \mathbb{N}$, and every domain size $N \in \mathbb{N}$,

$$|\delta_{\text{real}} - \delta_{\text{rand}}| \leq \epsilon_{\text{src-pr}}(\lambda, N, \epsilon_{\mathcal{D}}),$$

where

$$\begin{aligned} \delta_{\text{real}} &:= \Pr \left[\mathbf{A}(\text{crs}, y, \text{mk}\{k\}, k, \text{td}\{y\}) = 1 \mid \begin{array}{l} x \leftarrow \mathcal{D}_N \\ (\text{crs}, \text{mk}) \leftarrow \text{Gen}(1^\lambda, N) \\ k \leftarrow \{0, 1\}^{\ell(\lambda, \log N)} \\ y \leftarrow \Pi(\text{crs}, k, x) \\ \text{mk}\{k\} \leftarrow \text{PuncMK}(\text{mk}, k) \\ \text{td} \leftarrow \text{Trapdoor}(\text{mk}, k) \\ \text{td}\{y\} \leftarrow \text{PuncTD}(\text{td}, y) \end{array} \right] , \\ \delta_{\text{rand}} &:= \Pr \left[\mathbf{A}(\text{crs}, y, \text{mk}\{k\}, k, \text{td}\{y\}) = 1 \mid \begin{array}{l} (\text{crs}, \text{mk}) \leftarrow \text{Gen}(1^\lambda, N) \\ k \leftarrow \{0, 1\}^{\ell(\lambda, \log N)} \\ y \leftarrow [N] \\ \text{mk}\{k\} \leftarrow \text{PuncMK}(\text{mk}, k) \\ \text{td} \leftarrow \text{Trapdoor}(\text{mk}, k) \\ \text{td}\{y\} \leftarrow \text{PuncTD}(\text{td}, y) \end{array} \right] . \end{aligned}$$

Remark 6.4. For a uniform source, weak-source two-level punctured security is equivalent to security for an independently sampled uniform output, since $\Pi(\text{crs}, k, \cdot)$ is a permutation. This does not imply security for every fixed output. Instead, fixed-output two-level punctured security together with weak-source pseudorandomness implies weak-source two-level punctured security, as shown in Corollary 6.7.

Theorem 6.5. Fix $q(\cdot) = \text{polylog}(\cdot)$. Assume the existence of a secure indistinguishability obfuscation, a secure PPRP for the class of general swaps, and a secure injective length-doubling PRG. Then, there exists a trapdoor permutation with a full key domain in the CRS model satisfying two-level punctured security as defined in Definition 6.1, such that for every efficiently computable domain-size function $N = N(\lambda)$ that is exponential in λ , and every security parameter $\lambda \in \mathbb{N}$, it holds that, $\epsilon(\lambda, N) \leq \text{negl}(\lambda)$.

The running time of the scheme is $\text{poly}(\lambda, \log N, q(N))$.

Theorem 6.6 (Weak-source pseudorandomness). Fix $q(\cdot) = \text{polylog}(\cdot)$. Assume the existence of a secure indistinguishability obfuscation, a secure PPRP for the class of general swaps, and a secure injective length-doubling PRG. Then, the trapdoor permutation with a full key domain from Construction 6.8, instantiated with the base TDP from Construction 5.7, satisfies weak-source pseudorandomness as defined in Definition 6.3. More precisely, for every function $\epsilon_{\mathcal{D}} : \mathbb{N} \rightarrow [0, 1]$, every ensemble $\mathcal{D} := \{\mathcal{D}_N\}_{N \in \mathbb{N}}$ that is $(\epsilon_{\mathcal{D}}, q)$ -hard, every efficiently computable domain-size function $N = N(\lambda)$ that is exponential in λ , and every security parameter $\lambda \in \mathbb{N}$, it holds that

$$\epsilon_{\text{src-pr}}(\lambda, N, \epsilon_{\mathcal{D}}) \leq O(\epsilon_{\mathcal{D}}(N)) + \text{negl}(\lambda) .$$

The running time of the scheme is $\text{poly}(\lambda, \log N, q(N))$.

Corollary 6.7 (Weak-source two-level punctured security). Fix $q(\cdot) = \text{polylog}(\cdot)$. Assume there exists a trapdoor permutation with a full key domain in the CRS model that satisfies fixed-output two-level punctured security with error $\epsilon(\lambda, N)$, as in Definition 6.1, and a weak-source pseudorandomness with error $\epsilon_{\text{src-pr}}$, as in Definition 6.3. Then the same scheme satisfies weak-source two-level punctured security as in Definition 6.2. For every $(\epsilon_{\mathcal{D}}, q)$ -hard source ensemble $\mathcal{D}$ independent of setup and every efficiently computable $N = N(\lambda)$ exponential in λ ,

$$\epsilon_{\text{src}}(\lambda, N, \epsilon_{\mathcal{D}}) \leq \epsilon_{\text{src-pr}}(\lambda, N, \epsilon_{\mathcal{D}}) + \epsilon(\lambda, N).$$

In particular, under the assumptions of Theorems 6.5 and 6.6, with the concrete instantiations given there, Construction 6.8 satisfies

$$\epsilon_{\text{src}}(\lambda, N, \epsilon_{\mathcal{D}}) \leq O(\epsilon_{\mathcal{D}}(N)) + \text{negl}(\lambda).$$

6.2 Construction

Let $N \in \mathbb{N}$ and $\lambda \in \mathbb{N}$. Let $\text{PT} = (\text{PT.Gen}, \text{PT.II}, \text{PT.II}^{-1}, \text{PT.Punc})$ be a TDP with a puncturable trapdoor over $[N]$ as in Definition 5.1. Let $r = r(\lambda, \log N)$ be the number of random bits that PT.Gen uses, where r is polynomially bounded. For random bits $\rho \in \{0, 1\}^r$, we denote

$$(\text{pk}_{\rho}, \text{td}_{\rho}) := \text{PT.Gen}(1^{\lambda}, N; \rho).$$

For every such tape, pk_{ρ} defines a permutation of $[N]$, and td_{ρ} is its trapdoor.

Set $N' := 2^r$ and $\ell(\lambda, \log N) := r$. We identify $[N']$ with $\{0, 1\}^r$. Let

$$\text{punc} = (\text{punc}.\Pi, \text{punc}.\Pi^{-1}, \text{punc}.\text{Punc}, \text{punc}.\text{Eval}, \text{punc}.\text{Eval}^{-1})$$

be an outer puncturable PRP over $[N']$. The key K_{enc} encodes the randomness ρ used by $\text{PT}.\text{Gen}$:

$$K_{\text{enc}} := \text{punc}.\Pi(k_{\text{master}}, \rho), \quad \rho = \text{punc}.\Pi^{-1}(k_{\text{master}}, K_{\text{enc}}).$$

Every $K_{\text{enc}} \in \{0, 1\}^r$ decodes to randomness $\rho \in \{0, 1\}^r$ for $\text{PT}.\text{Gen}$. The outer puncturing algorithms are used only in the proof.

We use the following circuit families, with (λ, N) implicit.

$C_{\text{eval}}[k_{\text{master}}](K_{\text{enc}}, x)$:

1. Compute $\rho \leftarrow \text{punc}.\Pi^{-1}(k_{\text{master}}, K_{\text{enc}})$.
2. Compute $(\text{pk}_\rho, \text{td}_\rho) \leftarrow \text{PT}.\text{Gen}(1^\lambda, N; \rho)$.
3. Output $\text{PT}.\Pi(\text{pk}_\rho, x)$.

$C_{\text{derive}}[k_{\text{master}}][K_{\text{enc}}^* \rightarrow \perp](K_{\text{enc}})$:

1. If $K_{\text{enc}} = K_{\text{enc}}^*$, then output $\perp$.
2. Compute $\rho \leftarrow \text{punc}.\Pi^{-1}(k_{\text{master}}, K_{\text{enc}})$.
3. Compute $(\text{pk}_\rho, \text{td}_\rho) \leftarrow \text{PT}.\text{Gen}(1^\lambda, N; \rho)$ and output td_ρ .

Let $s := \text{poly}(\lambda, \log N, r)$ be large enough to upper-bound these circuits here and in the relevant hybrids in the proofs. This public padding bound is implicit below.

Construction 6.8. The trapdoor permutation with a full key domain is defined as follows.

- $\text{Gen}(1^\lambda, N)$:
 1. Sample $k_{\text{master}} \leftarrow \{0, 1\}^\lambda$.
 2. Compute $\widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{eval}}[k_{\text{master}}])$.
 3. Output (crs, mk) , where $\text{crs} := \widehat{C_{\text{eval}}}$ and $\text{mk} := k_{\text{master}}$.
- $\Pi(\text{crs}, K_{\text{enc}}, x)$: Parse crs as $\widehat{C_{\text{eval}}}$ and output $\widehat{C_{\text{eval}}}(K_{\text{enc}}, x)$.
- $\text{PuncMK}(\text{mk}, K_{\text{enc}}^*)$:
 1. Parse mk as k_{master} .
 2. Compute $\widehat{C_{\text{derive}}} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{derive}}[k_{\text{master}}][K_{\text{enc}}^* \rightarrow \perp])$.
 3. Output $\text{mk}\{K_{\text{enc}}^*\} := \widehat{C_{\text{derive}}}$.
- $\text{Trapdoor}(\text{mk}, K_{\text{enc}})$:
 1. If $\text{mk} = k_{\text{master}}$ was output by Gen , compute $\rho \leftarrow \text{punc}.\Pi^{-1}(k_{\text{master}}, K_{\text{enc}})$, compute $(\text{pk}_\rho, \text{td}_\rho) \leftarrow \text{PT}.\text{Gen}(1^\lambda, N; \rho)$, and output $\text{td} := \text{td}_\rho$.
 2. Else, if $\text{mk}\{K_{\text{enc}}^*\} = \widehat{C_{\text{derive}}}$ was output by PuncMK , output $\text{td} \leftarrow \widehat{C_{\text{derive}}}(K_{\text{enc}})$.
- $\text{PuncTD}(\text{td}, y^*)$: Output $\text{td}\{y^*\} \leftarrow \text{PT}.\text{Punc}(\text{td}, y^*)$.
- $\Pi^{-1}(\text{td}, y)$: Output $\text{PT}.\Pi^{-1}(\text{td}, y)$. The same algorithm handles an output-punctured trapdoor returned by PuncTD .

6.3 Proof of Theorem 6.5

Proof. Let PT , punc , and iO satisfy the assumptions of the theorem. Write ϵ_{PT} for the puncturable trapdoor security error of PT , $\epsilon_{\text{punc}}(\lambda, N')$ for the outer puncturable-PRP security error, and $\epsilon_{\text{iO}}(\lambda)$ for the obfuscation security error, with $N' = 2^r$. We prove correctness and then two-level punctured security.

Correctness. Fix any output $(\text{crs}, \text{mk}) = (\widehat{C_{\text{eval}}}, \text{k}_{\text{master}})$ of $\text{Gen}(1^\lambda, N)$. For every $K_{\text{enc}} \in \{0, 1\}^r$, define

$$\begin{aligned}\rho &:= \text{punc}.\Pi^{-1}(\text{k}_{\text{master}}, K_{\text{enc}}), \\ (\text{pk}_\rho, \text{td}_\rho) &:= \text{PT}.\text{Gen}(1^\lambda, N; \rho).\end{aligned}$$

Every key K_{enc} decodes to randomness $\rho \in \{0, 1\}^r$, which determines the pair $(\text{pk}_\rho, \text{td}_\rho)$ generated by $\text{PT}.\text{Gen}$. We verify the correctness properties as follows.

- *Full key-domain correctness.* By perfect functionality of the obfuscator, for every $x \in [N]$,

$$\begin{aligned}\Pi(\text{crs}, K_{\text{enc}}, x) &= \widehat{C_{\text{eval}}}(K_{\text{enc}}, x) \\ &= \text{PT}.\Pi(\text{pk}_\rho, x).\end{aligned}$$

Since $(\text{pk}_\rho, \text{td}_\rho)$ is an output of $\text{PT}.\text{Gen}$, the base TDP's functionality guarantees that this is a permutation of $[N]$. This holds for every $K_{\text{enc}} \in \{0, 1\}^r$, so the key domain is full. Evaluation is deterministic and efficient because the decoding, fixed-randomness key generation, and base evaluation are deterministic polynomial-time algorithms.

- *Trapdoor correctness.* The algorithm $\text{Trapdoor}(\text{mk}, K_{\text{enc}})$ deterministically returns td_ρ . By the base TDP's trapdoor correctness, for every $x \in [N]$,

$$\begin{aligned}\Pi^{-1}(\text{td}_\rho, \Pi(\text{crs}, K_{\text{enc}}, x)) \\ = \text{PT}.\Pi^{-1}(\text{td}_\rho, \text{PT}.\Pi(\text{pk}_\rho, x)) = x.\end{aligned}$$

- *Punctured key correctness and consistency.* Fix $K_{\text{enc}}^* \in \{0, 1\}^r$ and let $\text{mk}\{K_{\text{enc}}^*\} \leftarrow \text{PuncMK}(\text{mk}, K_{\text{enc}}^*)$. For every $K_{\text{enc}} \neq K_{\text{enc}}^*$, the punctured derivation circuit decodes the same randomness ρ and runs the same fixed-randomness key generation as the unpunctured derivation algorithm. Perfect functionality of the obfuscator therefore gives

$$\text{Trapdoor}(\text{mk}\{K_{\text{enc}}^*\}, K_{\text{enc}}) = \text{td}_\rho = \text{Trapdoor}(\text{mk}, K_{\text{enc}}).$$

This is equality of the trapdoors themselves, not just their functionality. Inversion correctness then follows from the previous item. At $K_{\text{enc}} = K_{\text{enc}}^*$, the punctured derivation circuit returns $\perp$.

- *Punctured trapdoor correctness.* Fix $y^* \in [N]$ and sample

$$\text{td}_\rho\{y^*\} \leftarrow \text{PT}.\text{Punc}(\text{td}_\rho, y^*).$$

This is exactly the output distribution of $\text{PuncTD}(\text{td}_\rho, y^*)$. For every $y \neq y^*$, the base TDP's punctured trapdoor correctness gives, with probability one,

$$\begin{aligned}\Pi(\text{crs}, K_{\text{enc}}, \Pi^{-1}(\text{td}_\rho\{y^*\}, y)) \\ = \text{PT}.\Pi(\text{pk}_\rho, \text{PT}.\Pi^{-1}(\text{td}_\rho\{y^*\}, y)) = y.\end{aligned}$$

The rejection requirement in Definition 5.1 also gives, with probability one,

$$\text{PT}.\Pi^{-1}(\text{td}_\rho\{y^*\}, y^*) = \perp.$$

Therefore, $\Pi^{-1}(\text{td}_\rho\{y^*\}, y^*) = \perp$, as required.

Two-level punctured security. Under the assumptions of Theorem 6.5, we get that Theorem 5.4 provides the base TDP PT with a puncturable trapdoor, and Theorem 4.2 provides the outer puncturable PRP over $[N']$, where $N' = 2^r$. Let $\epsilon_{\text{PT}}(\lambda, N)$, $\epsilon_{\text{punc}}(\lambda, N')$, and $\epsilon_{\text{iO}}(\lambda)$ denote the security errors of PT, the outer puncturable PRP, and the indistinguishability obfuscator, respectively.

The proof punctures the outer key, replaces the decoded randomness ρ^* by an independent uniform random bits, and restores the outer key. We then use iO to replace the selected branch's computation of PT.Gen from ρ^* by the resulting public key pk^* . This permits a black-box reduction to the puncturable trapdoor security of PT.

Fix $y^* \in [N]$ independently of setup and the sampled encoded key, and a PPT adversary **A**. The challenge y^* remains fixed in every hybrid. For $i \in \{0, \dots, 5\}$, let δ_i be the probability that $\text{PT}.\Pi(\text{pk}^*, \hat{x}) = y^*$, where pk^* is the selected base public key in that hybrid. Invalid outputs are treated as failures. This event can be checked by public evaluation without knowing the preimage of y^* .

Hybrid 0: The real experiment. The CRS is an obfuscation of the evaluation circuit, and the punctured master key is an obfuscation of the derivation circuit below.

$C_{\text{eval}}[\text{k}_{\text{master}}](K_{\text{enc}}, x)$:

1. Compute $\rho \leftarrow \text{punc}.\Pi^{-1}(\text{k}_{\text{master}}, K_{\text{enc}})$.
2. Compute $(\text{pk}_\rho, \text{td}_\rho) \leftarrow \text{PT}.\text{Gen}(1^\lambda, N; \rho)$.
3. Output $\text{PT}.\Pi(\text{pk}_\rho, x)$.

$C_{\text{derive}}[\text{k}_{\text{master}}][K_{\text{enc}}^* \rightarrow \perp](K_{\text{enc}})$:

1. If $K_{\text{enc}} = K_{\text{enc}}^*$, output $\perp$.
2. Else, compute $\rho \leftarrow \text{punc}.\Pi^{-1}(\text{k}_{\text{master}}, K_{\text{enc}})$.
3. Compute $(\text{pk}_\rho, \text{td}_\rho) \leftarrow \text{PT}.\text{Gen}(1^\lambda, N; \rho)$ and output td_ρ .

Here, we sample K_{enc}^* before generating the CRS. This only reorders the sampling steps for independent samples. The full experiment is

$$\delta_0 := \Pr \left[\text{PT}.\Pi(\text{pk}^*, \hat{x}) = y^* \mid \begin{array}{l} \text{k}_{\text{master}} \leftarrow \{0, 1\}^\lambda \\ K_{\text{enc}}^* \leftarrow \{0, 1\}^r \\ \rho^* \leftarrow \text{punc}.\Pi^{-1}(\text{k}_{\text{master}}, K_{\text{enc}}^*) \\ (\text{pk}^*, \text{td}^*) \leftarrow \text{PT}.\text{Gen}(1^\lambda, N; \rho^*) \\ \text{crs} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{eval}}[\text{k}_{\text{master}}]) \\ \widehat{C_{\text{derive}}} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{derive}}[\text{k}_{\text{master}}][K_{\text{enc}}^* \rightarrow \perp]) \\ \text{td}^*\{y^*\} \leftarrow \text{PT}.\text{Punc}(\text{td}^*, y^*) \\ \hat{x} \leftarrow \mathbf{A}(\text{crs}, y^*, \widehat{C_{\text{derive}}}, K_{\text{enc}}^*, \text{td}^*\{y^*\}) \end{array} \right].$$

Hybrid 1: Isolating the selected key. In this hybrid, we replace the CRS by an obfuscation of the following circuit:

$C_{\text{eval}}[\mathbf{k}_{\text{master}}, \rho^*, K_{\text{enc}}^*](K_{\text{enc}}, x)$:

1. If $K_{\text{enc}} = K_{\text{enc}}^*$, set $\rho := \rho^*$; otherwise, set $\rho := \text{punc}.\Pi^{-1}(\mathbf{k}_{\text{master}}, K_{\text{enc}})$.
2. Compute $(\text{pk}_\rho, \text{td}_\rho) \leftarrow \text{PT.Gen}(1^\lambda, N; \rho)$.
3. Output $\text{PT}.\Pi(\text{pk}_\rho, x)$.

We do not hardwire ρ^* into C_{derive} , since this circuit rejects K_{enc}^* and therefore never uses the corresponding randomness. Since $\rho^* = \text{punc}.\Pi^{-1}(\mathbf{k}_{\text{master}}, K_{\text{enc}}^*)$, this circuit is functionally equivalent to the original evaluation circuit on every input. The rest of the experiment is unchanged. Therefore, by iO security,

$$|\delta_1 - \delta_0| \leq \epsilon_{\text{iO}}(\lambda). \quad (5)$$

The full experiment is

$$\delta_1 := \Pr \left[\text{PT}.\Pi(\text{pk}^*, \hat{x}) = y^* \mid \begin{array}{l} \mathbf{k}_{\text{master}} \leftarrow \{0, 1\}^\lambda \\ K_{\text{enc}}^* \leftarrow \{0, 1\}^r \\ \rho^* \leftarrow \text{punc}.\Pi^{-1}(\mathbf{k}_{\text{master}}, K_{\text{enc}}^*) \\ (\text{pk}^*, \text{td}^*) \leftarrow \text{PT.Gen}(1^\lambda, N; \rho^*) \\ \text{crs} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{eval}}[\mathbf{k}_{\text{master}}, \rho^*, K_{\text{enc}}^*]) \\ \widehat{C_{\text{derive}}} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{derive}}[\mathbf{k}_{\text{master}}][K_{\text{enc}}^* \rightarrow \perp]) \\ \text{td}^*\{y^*\} \leftarrow \text{PT.Punc}(\text{td}^*, y^*) \\ \hat{x} \leftarrow \mathbf{A}(\text{crs}, y^*, \widehat{C_{\text{derive}}}, K_{\text{enc}}^*, \text{td}^*\{y^*\}) \end{array} \right].$$

Hybrid 2: Puncturing the outer key. Sample

$$\mathbf{k}_{\text{master}}^{\text{punc}} \leftarrow \text{punc.Punc}(\mathbf{k}_{\text{master}}, \rho^*, K_{\text{enc}}^*).$$

We replace the outer master key by its punctured version in both obfuscated circuits. The new circuits are

$C_{\text{eval}}[\mathbf{k}_{\text{master}}^{\text{punc}}, \rho^*, K_{\text{enc}}^*](K_{\text{enc}}, x)$:

1. If $K_{\text{enc}} = K_{\text{enc}}^*$, set $\rho := \rho^*$.
2. Else, compute $\rho \leftarrow \text{punc.Eval}^{-1}(\mathbf{k}_{\text{master}}^{\text{punc}}, K_{\text{enc}})$.
3. Compute $(\text{pk}_\rho, \text{td}_\rho) \leftarrow \text{PT.Gen}(1^\lambda, N; \rho)$.
4. Output $\text{PT}.\Pi(\text{pk}_\rho, x)$.

$C_{\text{derive}}[\mathbf{k}_{\text{master}}^{\text{punc}}][K_{\text{enc}}^* \rightarrow \perp](K_{\text{enc}})$:

1. If $K_{\text{enc}} = K_{\text{enc}}^*$, output $\perp$.
2. Else, compute $\rho \leftarrow \text{punc.Eval}^{-1}(\mathbf{k}_{\text{master}}^{\text{punc}}, K_{\text{enc}})$.
3. Compute $(\text{pk}_\rho, \text{td}_\rho) \leftarrow \text{PT.Gen}(1^\lambda, N; \rho)$ and output td_ρ .

Neither circuit evaluates on the punctured output. Outer puncturing correctness therefore makes each circuit functionally equivalent to its predecessor. Replacing the two obfuscations one at a time gives

$$|\delta_2 - \delta_1| \leq 2\epsilon_{\text{iO}}(\lambda). \quad (6)$$

The full experiment is

$$\delta_2 := \Pr \left[\text{PT}.\Pi(\text{pk}^*, \hat{x}) = y^* \mid \begin{array}{l} k_{\text{master}} \leftarrow \{0, 1\}^\lambda \\ K_{\text{enc}}^* \leftarrow \{0, 1\}^r \\ \rho^* \leftarrow \text{punc}.\Pi^{-1}(k_{\text{master}}, K_{\text{enc}}^*) \\ (\text{pk}^*, \text{td}^*) \leftarrow \text{PT}.\text{Gen}(1^\lambda, N; \rho^*) \\ k_{\text{master}}^{\text{punc}} \leftarrow \text{punc}.\text{Punc}(k_{\text{master}}, \rho^*, K_{\text{enc}}^*) \\ \text{crs} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{eval}}[k_{\text{master}}^{\text{punc}}, \rho^*, K_{\text{enc}}^*]) \\ \widehat{C_{\text{derive}}} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{derive}}[k_{\text{master}}^{\text{punc}}][K_{\text{enc}}^* \rightarrow \perp]) \\ \text{td}^*\{y^*\} \leftarrow \text{PT}.\text{Punc}(\text{td}^*, y^*) \\ \hat{x} \leftarrow \mathbf{A}(\text{crs}, y^*, \widehat{C_{\text{derive}}}, K_{\text{enc}}^*, \text{td}^*\{y^*\}) \end{array} \right].$$

Hybrid 3: Decoupling the randomness from the encoded key. Replace the decoded value $\rho^* = \text{punc}.\Pi^{-1}(k_{\text{master}}, K_{\text{enc}}^*)$ by an independent uniform $\rho^* \leftarrow \{0, 1\}^r$. The rest of the experiment uses this newly sampled ρ^* . For the fixed challenge y^* , the adversary's view can be generated from $(\rho^*, k_{\text{master}}^{\text{punc}}, K_{\text{enc}}^*)$, where $\rho^*, K_{\text{enc}}^* \leftarrow \{0, 1\}^r$ are sampled independently, $k_{\text{master}} \leftarrow \{0, 1\}^\lambda$, and $k_{\text{master}}^{\text{punc}} \leftarrow \text{punc}.\text{Punc}(k_{\text{master}}, \rho^*, K_{\text{enc}}^*)$.

This is exactly an application of output-puncturing security of the outer PRP. For a fixed K_{enc}^* , its experiment provides $(\rho^*, k_{\text{master}}^{\text{punc}}, K_{\text{enc}}^*)$, where ρ^* is either the actual preimage or an independent uniform string, with the outer key punctured at the corresponding pair in each case. Given this tuple, the reduction runs $\text{PT}.\text{Gen}(1^\lambda, N; \rho^*)$ to obtain $(\text{pk}^*, \text{td}^*)$, generates both obfuscations, and punctures td^* at the fixed y^* . It runs $\mathbf{A}$ and tests whether $\text{PT}.\Pi(\text{pk}^*, \hat{x}) = y^*$. Hence

$$|\delta_3 - \delta_2| \leq \epsilon_{\text{punc}}(\lambda, N'). \quad (7)$$

We prove the bound for each fixed encoded key K_{enc}^* , hardwired into the distinguisher, and then average over $K_{\text{enc}}^* \leftarrow \{0, 1\}^r$. The output y^* is fixed throughout this argument. The obfuscation reductions likewise generate the selected base public key and check success by public evaluation.

The full experiment is

$$\delta_3 := \Pr \left[\text{PT}.\Pi(\text{pk}^*, \hat{x}) = y^* \mid \begin{array}{l} k_{\text{master}} \leftarrow \{0, 1\}^\lambda \\ K_{\text{enc}}^* \leftarrow \{0, 1\}^r \\ \rho^* \leftarrow \{0, 1\}^r \\ (\text{pk}^*, \text{td}^*) \leftarrow \text{PT}.\text{Gen}(1^\lambda, N; \rho^*) \\ k_{\text{master}}^{\text{punc}} \leftarrow \text{punc}.\text{Punc}(k_{\text{master}}, \rho^*, K_{\text{enc}}^*) \\ \text{crs} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{eval}}[k_{\text{master}}^{\text{punc}}, \rho^*, K_{\text{enc}}^*]) \\ \widehat{C_{\text{derive}}} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{derive}}[k_{\text{master}}^{\text{punc}}][K_{\text{enc}}^* \rightarrow \perp]) \\ \text{td}^*\{y^*\} \leftarrow \text{PT}.\text{Punc}(\text{td}^*, y^*) \\ \hat{x} \leftarrow \mathbf{A}(\text{crs}, y^*, \widehat{C_{\text{derive}}}, K_{\text{enc}}^*, \text{td}^*\{y^*\}) \end{array} \right].$$

Hybrid 4: Restoring the outer key. In this hybrid, we restore the unpunctured outer key in both circuits, retaining the selected branch and the rejection at K_{enc}^* . The new circuits are

$C_{\text{eval}}[\textcolor{red}{k}_{\text{master}}, \rho^*, K_{\text{enc}}^*](K_{\text{enc}}, x)$:

1. If $K_{\text{enc}} = K_{\text{enc}}^*$, set $\rho := \rho^*$.
2. Else, compute $\rho \leftarrow \text{punc}.\Pi^{-1}(\textcolor{red}{k}_{\text{master}}, K_{\text{enc}})$.

3. Compute $(\text{pk}_\rho, \text{td}_\rho) \leftarrow \text{PT.Gen}(1^\lambda, N; \rho)$.
4. Output $\text{PT.II}(\text{pk}_\rho, x)$.

$C_{\text{derive}}[\text{k}_{\text{master}}][K_{\text{enc}}^* \rightarrow \perp](K_{\text{enc}})$:

1. If $K_{\text{enc}} = K_{\text{enc}}^*$, output $\perp$.
2. Else, compute $\rho \leftarrow \text{punc.II}^{-1}(\text{k}_{\text{master}}, K_{\text{enc}})$.
3. Compute $(\text{pk}_\rho, \text{td}_\rho) \leftarrow \text{PT.Gen}(1^\lambda, N; \rho)$ and output td_ρ .

For every $K_{\text{enc}} \neq K_{\text{enc}}^*$, outer puncturing correctness guarantees that decoding with $\text{k}_{\text{master}}^{\text{punc}}$ returns exactly the same randomness as decoding with k_{master} . At K_{enc}^* , neither circuit invokes decoding: the evaluation circuit uses the hardwired ρ^* , and the derivation circuit returns $\perp$. Hence, restoring k_{master} preserves both circuits' functionality. Thus, two applications of iO give

$$|\delta_4 - \delta_3| \leq 2\epsilon_{\text{iO}}(\lambda). \quad (8)$$

The punctured outer key is no longer used anywhere in the view. In particular, the selected randomness ρ^* is independent of both k_{master} and K_{enc}^* . The derivation circuit uses only these latter two values.

The full experiment is

$$\delta_4 := \Pr \left[\text{PT.II}(\text{pk}^*, \hat{x}) = y^* \mid \begin{array}{l} \text{k}_{\text{master}} \leftarrow \{0, 1\}^\lambda \\ K_{\text{enc}}^* \leftarrow \{0, 1\}^r \\ \rho^* \leftarrow \{0, 1\}^r \\ (\text{pk}^*, \text{td}^*) \leftarrow \text{PT.Gen}(1^\lambda, N; \rho^*) \\ \text{crs} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{eval}}[\text{k}_{\text{master}}, \rho^*, K_{\text{enc}}^*]) \\ \widehat{C_{\text{derive}}} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{derive}}[\text{k}_{\text{master}}][K_{\text{enc}}^* \rightarrow \perp]) \\ \text{td}^*\{y^*\} \leftarrow \text{PT.Punc}(\text{td}^*, y^*) \\ \hat{x} \leftarrow \mathbf{A}(\text{crs}, y^*, \widehat{C_{\text{derive}}}, K_{\text{enc}}^*, \text{td}^*\{y^*\}) \end{array} \right].$$

Hybrid 5: Embedding the selected public key. Compute $(\text{pk}^*, \text{td}^*) = \text{PT.Gen}(1^\lambda, N; \rho^*)$ as before. Replace the CRS circuit by

$C_{\text{eval}}^{\text{pk}}[\text{k}_{\text{master}}, \text{pk}^*, K_{\text{enc}}^*](K_{\text{enc}}, x)$:

1. If $K_{\text{enc}} = K_{\text{enc}}^*$, output $\text{PT.II}(\text{pk}^*, x)$.
2. Otherwise, compute $\rho \leftarrow \text{punc.II}^{-1}(\text{k}_{\text{master}}, K_{\text{enc}})$ and $(\text{pk}_\rho, \text{td}_\rho) \leftarrow \text{PT.Gen}(1^\lambda, N; \rho)$, and output $\text{PT.II}(\text{pk}_\rho, x)$.

The circuit is functionally equivalent to that in Hybrid 4: at the selected key, it uses exactly the public key that the previous circuit recomputed from ρ^* . Elsewhere, the computation remains unchanged. The selected punctured trapdoor and the punctured master key are unchanged. Therefore,

$$|\delta_5 - \delta_4| \leq \epsilon_{\text{iO}}(\lambda). \quad (9)$$

The full experiment is

$$\delta_5 := \Pr \left[\text{PT}.\Pi(\text{pk}^*, \hat{x}) = y^* \mid \begin{array}{l} \mathbf{k}_{\text{master}} \leftarrow \{0, 1\}^\lambda \\ K_{\text{enc}}^* \leftarrow \{0, 1\}^r \\ \rho^* \leftarrow \{0, 1\}^r \\ (\text{pk}^*, \text{td}^*) \leftarrow \text{PT}.\text{Gen}(1^\lambda, N; \rho^*) \\ \text{crs} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{eval}}^{\text{pk}}[\mathbf{k}_{\text{master}}, \text{pk}^*, K_{\text{enc}}^*]) \\ \widehat{C_{\text{derive}}} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{derive}}[\mathbf{k}_{\text{master}}][K_{\text{enc}}^* \rightarrow \perp]) \\ \text{td}^*\{y^*\} \leftarrow \text{PT}.\text{Punc}(\text{td}^*, y^*) \\ \hat{x} \leftarrow \mathbf{A}(\text{crs}, y^*, \widehat{C_{\text{derive}}}, K_{\text{enc}}^*, \text{td}^*\{y^*\}) \end{array} \right].$$

Reduction to PT. We now reduce Hybrid 5 to the puncturable trapdoor security of the base TDP in Definition 5.1, established by Theorem 5.4. Given a base challenge $(\text{pk}^*, \text{td}^*\{y^*\}, y^*)$, the reduction $\mathbf{A}'$ proceeds as follows:

1. Sample $\mathbf{k}_{\text{master}} \leftarrow \{0, 1\}^\lambda$ and $K_{\text{enc}}^* \leftarrow \{0, 1\}^r$ independently.
2. Generate $\text{crs} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{eval}}^{\text{pk}}[\mathbf{k}_{\text{master}}, \text{pk}^*, K_{\text{enc}}^*])$.
3. Generate $\text{mk}\{K_{\text{enc}}^*\} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{derive}}[\mathbf{k}_{\text{master}}][K_{\text{enc}}^* \rightarrow \perp])$.
4. Run $\hat{x} \leftarrow \mathbf{A}(\text{crs}, y^*, \text{mk}\{K_{\text{enc}}^*\}, K_{\text{enc}}^*, \text{td}^*\{y^*\})$ and output $\hat{x}$.

In Hybrid 5, uniform ρ^* generates an honest base key pair independently of $\mathbf{k}_{\text{master}}$ and K_{enc}^* . The selected branch of the CRS uses only pk^* . The derivation circuit rejects the selected key. Consequently, the reduction simulates Hybrid 5 exactly without knowing ρ^* or the full trapdoor td^* . Its success probability is

$$\begin{aligned} \delta_5 &= \Pr \left[\text{PT}.\Pi(\text{pk}^*, \hat{x}) = y^* \mid \begin{array}{l} (\text{pk}^*, \text{td}^*) \leftarrow \text{PT}.\text{Gen}(1^\lambda, N) \\ \text{td}^*\{y^*\} \leftarrow \text{PT}.\text{Punc}(\text{td}^*, y^*) \\ \hat{x} \leftarrow \mathbf{A}'(\text{pk}^*, \text{td}^*\{y^*\}, y^*) \end{array} \right] \\ &\leq \epsilon_{\text{PT}}(\lambda, N). \end{aligned}$$

This uses puncturable trapdoor security for an output fixed independently of key generation.

Finishing the proof. Summing the obfuscation losses and the outer puncturing loss gives the black-box guarantee

$$\delta_0 \leq \epsilon_{\text{PT}}(\lambda, N) + O(\epsilon_{\text{iO}}(\lambda)) + \epsilon_{\text{punc}}(\lambda, N').$$

For the instantiation of PT in Theorem 5.4, the updated puncturable-PRP proof gives

$$\epsilon_{\text{PT}}(\lambda, N) \leq O\left(\frac{1}{N} + \epsilon_{\text{PRG}}(N) + \epsilon_{\text{iO}}(\lambda) + \epsilon_{\text{PPRP}}(\lambda, N)\right).$$

Apply Theorem 4.2 to the outer permutation over $[N']$ as well. We obtain

$$\begin{aligned} \delta_0 &\leq O\left(\frac{1}{N} + \frac{1}{N'} + \epsilon_{\text{PRG}}(N) + \epsilon_{\text{PRG}}(N') + \epsilon_{\text{iO}}(\lambda)\right) \\ &\quad + O(\epsilon_{\text{PPRP}}(\lambda, N) + \epsilon_{\text{PPRP}}(\lambda, N')). \end{aligned}$$

For the concrete base construction, $r \geq \lambda$ and r is polynomial in $(\lambda, \log N)$. Thus both N and $N' = 2^r$ are admissible exponential domains, and security of the underlying primitives makes the bound negligible: $\delta_0 \leq \text{negl}(\lambda)$, as required. $\square$

6.4 Proof of Theorem 6.6

Proof. Under the assumptions of Theorem 6.6, Theorem 5.5 provides the base TDP PT with a puncturable trapdoor and weak-source pseudorandomness, and Theorem 4.2 provides the outer puncturable PRP over $[N']$, where $N' = 2^r$. Let $\epsilon_{\text{PT}, \text{src-pr}}(\lambda, N, \epsilon_{\mathcal{D}})$, $\epsilon_{\text{punc}}(\lambda, N')$, and $\epsilon_{\text{iO}}(\lambda)$ denote the weak-source pseudorandomness error of PT, the security error of the outer puncturable PRP, and the security error of the indistinguishability obfuscator, respectively. Let $\mathcal{D} := \{\mathcal{D}_N\}_{N \in \mathbb{N}}$ be an $(\epsilon_{\mathcal{D}}, q)$ -hard ensemble, and let $\mathbf{A}$ be a PPT distinguisher. Define

$$\delta_{\text{real}} := \Pr \left[\mathbf{A}(\text{crs}, y^*, \text{mk}\{K_{\text{enc}}^*\}, K_{\text{enc}}^*, \text{td}\{y^*\}) = 1 \mid \begin{array}{l} x^* \leftarrow \mathcal{D}_N \\ (\text{crs}, \text{mk}) \leftarrow \text{Gen}(1^\lambda, N) \\ K_{\text{enc}}^* \leftarrow \{0, 1\}^{\ell(\lambda, \log N)} \\ y^* \leftarrow \Pi(\text{crs}, K_{\text{enc}}^*, x^*) \\ \text{mk}\{K_{\text{enc}}^*\} \leftarrow \text{PuncMK}(\text{mk}, K_{\text{enc}}^*) \\ \text{td} \leftarrow \text{Trapdoor}(\text{mk}, K_{\text{enc}}^*) \\ \text{td}\{y^*\} \leftarrow \text{PuncTD}(\text{td}, y^*) \end{array} \right].$$

Similarly, define

$$\delta_{\text{rand}} := \Pr \left[\mathbf{A}(\text{crs}, y^*, \text{mk}\{K_{\text{enc}}^*\}, K_{\text{enc}}^*, \text{td}\{y^*\}) = 1 \mid \begin{array}{l} (\text{crs}, \text{mk}) \leftarrow \text{Gen}(1^\lambda, N) \\ K_{\text{enc}}^* \leftarrow \{0, 1\}^{\ell(\lambda, \log N)} \\ y^* \leftarrow [N] \\ \text{mk}\{K_{\text{enc}}^*\} \leftarrow \text{PuncMK}(\text{mk}, K_{\text{enc}}^*) \\ \text{td} \leftarrow \text{Trapdoor}(\text{mk}, K_{\text{enc}}^*) \\ \text{td}\{y^*\} \leftarrow \text{PuncTD}(\text{td}, y^*) \end{array} \right].$$

Apply Hybrids 0–5 of the preceding proof to each experiment, measuring acceptance rather than inversion success. Instead of fixing the challenge output, in the real experiment sample $x^* \leftarrow \mathcal{D}_N$ and set $y^* = \text{PT}.\Pi(\text{pk}^*, x^*)$. For the random experiment, sample y^* uniformly and independently of key generation. In both cases, generate the selected punctured trapdoor using $\text{PT}.\text{Punc}(\text{td}^*, y^*)$. The circuit-equivalence arguments are unchanged. In the outer puncturing transition, the reduction knows ρ^* from its challenge tuple, so it can run $\text{PT}.\text{Gen}$, generate the selected punctured trapdoor, and simulate both obfuscations. For the real experiment, analyze each reduction for a fixed source point supplied as non-uniform advice and then average. This does not require an efficient sampler for $\mathcal{D}$.

Write $\widehat{\delta}_{\text{real}}$ and $\widehat{\delta}_{\text{rand}}$ for the two acceptance probabilities after these transitions. Then

$$\begin{aligned} |\delta_{\text{real}} - \widehat{\delta}_{\text{real}}| &\leq O(\epsilon_{\text{iO}}(\lambda)) + \epsilon_{\text{punc}}(\lambda, N'), \\ |\delta_{\text{rand}} - \widehat{\delta}_{\text{rand}}| &\leq (\epsilon_{\text{iO}}(\lambda)) + \epsilon_{\text{punc}}(\lambda, N'). \end{aligned}$$

At this point, the selected base key pair is generated with uniform randomness independent of the outer master key and encoded key, and its public key is embedded in the selected branch of $C_{\text{eval}}^{\text{pk}}$.

Construct a distinguisher $\mathbf{A}'$ for the weak-source pseudorandomness of PT. Given $(\text{pk}^*, \text{td}^*\{y^*\}, y^*)$, it:

1. Samples $\text{k}_{\text{master}} \leftarrow \{0, 1\}^\lambda$ and $K_{\text{enc}}^* \leftarrow \{0, 1\}^r$ independently.
2. Generates $\text{crs} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{eval}}^{\text{pk}}[\text{k}_{\text{master}}, \text{pk}^*, K_{\text{enc}}^*])$.
3. Generates $\text{mk}\{K_{\text{enc}}^*\} \leftarrow \text{iO}(1^\lambda, 1^s, C_{\text{derive}}[\text{k}_{\text{master}}][K_{\text{enc}}^* \rightarrow \perp])$.

4. Runs $\mathbf{A}(\text{crs}, y^*, \text{mk}\{K_{\text{enc}}^*\}, K_{\text{enc}}^*, \text{td}^*\{y^*\})$ and outputs its answer.

This simulation needs neither ρ^* nor the full selected trapdoor. For every other encoded key, the reduction can run PT.Gen on the decoded randomness itself. Consequently, its two acceptance probabilities are exactly

$$\begin{aligned}\widehat{\delta}_{\text{real}} &= \Pr \left[\mathbf{A}'(\text{pk}^*, \text{td}^*\{y^*\}, y^*) = 1 \mid \begin{array}{l} x^* \leftarrow \mathcal{D}_N \\ (\text{pk}^*, \text{td}^*) \leftarrow \text{PT.Gen}(1^\lambda, N) \\ y^* \leftarrow \text{PT.II}(\text{pk}^*, x^*) \\ \text{td}^*\{y^*\} \leftarrow \text{PT.Punc}(\text{td}^*, y^*) \end{array} \right], \\ \widehat{\delta}_{\text{rand}} &= \Pr \left[\mathbf{A}'(\text{pk}^*, \text{td}^*\{y^*\}, y^*) = 1 \mid \begin{array}{l} (\text{pk}^*, \text{td}^*) \leftarrow \text{PT.Gen}(1^\lambda, N) \\ y^* \leftarrow [N] \\ \text{td}^*\{y^*\} \leftarrow \text{PT.Punc}(\text{td}^*, y^*) \end{array} \right].\end{aligned}$$

By the assumed security of PT ,

$$|\widehat{\delta}_{\text{real}} - \widehat{\delta}_{\text{rand}}| \leq \epsilon_{\text{PT,src-pr}}(\lambda, N, \epsilon_{\mathcal{D}}).$$

The triangle inequality now gives

$$\begin{aligned}|\delta_{\text{real}} - \delta_{\text{rand}}| &\leq \epsilon_{\text{PT,src-pr}}(\lambda, N, \epsilon_{\mathcal{D}}) \\ &\quad + 12\epsilon_{\text{iO}}(\lambda) + 2\epsilon_{\text{punc}}(\lambda, N').\end{aligned}$$

For the concrete instantiation, use Theorem 5.5 for PT and Theorem 4.2 for the outer permutation. The base key-generation randomness has length $r \geq \lambda$, polynomial in $(\lambda, \log N, q(N))$, so $N' = 2^r$ is an admissible exponential domain. Substituting the two security bounds gives $O(\epsilon_{\mathcal{D}}(N)) + \text{negl}(\lambda)$, as required. $\square$

6.5 Proof of Corollary 6.7

Proof. Assume the CRS scheme has two-level punctured security error ϵ and weak-source pseudo-randomness error $\epsilon_{\text{src-pr}}$, as in the corollary. The implication below applies to any such scheme; the instantiation through PT is used only at the end. Fix an $(\epsilon_{\mathcal{D}}, q)$ -hard source ensemble $\mathcal{D}$ independent of setup, an admissible domain-size function $N = N(\lambda)$, and a PPT inverter $\mathbf{A}$. Its weak-source success probability is

$$p_{\text{src}} := \Pr \left[\widehat{x} = x^* \mid \begin{array}{l} x^* \leftarrow \mathcal{D}_N \\ (\text{crs}, \text{mk}) \leftarrow \text{Gen}(1^\lambda, N) \\ \mathbf{k} \leftarrow \{0, 1\}^{\ell(\lambda, \log N)} \\ y^* \leftarrow \Pi(\text{crs}, \mathbf{k}, x^*) \\ \text{mk}\{\mathbf{k}\} \leftarrow \text{PuncMK}(\text{mk}, \mathbf{k}) \\ \text{td} \leftarrow \text{Trapdoor}(\text{mk}, \mathbf{k}) \\ \text{td}\{y^*\} \leftarrow \text{PuncTD}(\text{td}, y^*) \\ \widehat{x} \leftarrow \mathbf{A}(\text{crs}, y^*, \text{mk}\{\mathbf{k}\}, \mathbf{k}, \text{td}\{y^*\}) \end{array} \right].$$

Construct a distinguisher $\mathbf{A}'$ for Definition 6.3. On input $(\text{crs}, y^*, \text{mk}\{\mathbf{k}\}, \mathbf{k}, \text{td}\{y^*\})$:

1. Run $\widehat{x} \leftarrow \mathbf{A}(\text{crs}, y^*, \text{mk}\{\mathbf{k}\}, \mathbf{k}, \text{td}\{y^*\})$.
2. If $\widehat{x} \notin [N]$, output 0.

3. Output 1 if $\Pi(\text{crs}, k, \hat{x}) = y^*$, and output 0 otherwise.

Write δ_{real} and δ_{rand} for its acceptance probabilities. Explicitly,

$$\delta_{\text{real}} := \Pr \left[\mathbf{A}'(\text{crs}, y^*, \text{mk}\{k\}, k, \text{td}\{y^*\}) = 1 \mid \begin{array}{l} x^* \leftarrow \mathcal{D}_N \\ (\text{crs}, \text{mk}) \leftarrow \text{Gen}(1^\lambda, N) \\ k \leftarrow \{0, 1\}^{\ell(\lambda, \log N)} \\ y^* \leftarrow \Pi(\text{crs}, k, x^*) \\ \text{mk}\{k\} \leftarrow \text{PuncMK}(\text{mk}, k) \\ \text{td} \leftarrow \text{Trapdoor}(\text{mk}, k) \\ \text{td}\{y^*\} \leftarrow \text{PuncTD}(\text{td}, y^*) \end{array} \right],$$

and

$$\delta_{\text{rand}} := \Pr \left[\mathbf{A}'(\text{crs}, y^*, \text{mk}\{k\}, k, \text{td}\{y^*\}) = 1 \mid \begin{array}{l} (\text{crs}, \text{mk}) \leftarrow \text{Gen}(1^\lambda, N) \\ k \leftarrow \{0, 1\}^{\ell(\lambda, \log N)} \\ y^* \leftarrow [N] \\ \text{mk}\{k\} \leftarrow \text{PuncMK}(\text{mk}, k) \\ \text{td} \leftarrow \text{Trapdoor}(\text{mk}, k) \\ \text{td}\{y^*\} \leftarrow \text{PuncTD}(\text{td}, y^*) \end{array} \right].$$

For every honestly generated CRS and every valid key, evaluation is a permutation. In the real experiment,

$$\begin{aligned} \Pi(\text{crs}, k, \hat{x}) = y^* &\iff \Pi(\text{crs}, k, \hat{x}) = \Pi(\text{crs}, k, x^*) \\ &\iff \hat{x} = x^*. \end{aligned}$$

Invalid outputs are rejected. Thus the acceptance event of $\mathbf{A}'$ is exactly the recovery event of $\mathbf{A}$, under the same sampling distribution. Consequently,

$$p_{\text{src}} = \delta_{\text{real}} \leq \delta_{\text{rand}} + \epsilon_{\text{src-pr}}(\lambda, N, \epsilon_{\mathcal{D}}).$$

For each fixed $y \in [N]$, define

$$p_y := \Pr \left[\begin{array}{l} \hat{x} \in [N] \\ \wedge \Pi(\text{crs}, k, \hat{x}) = y \end{array} \mid \begin{array}{l} (\text{crs}, \text{mk}) \leftarrow \text{Gen}(1^\lambda, N) \\ k \leftarrow \{0, 1\}^{\ell(\lambda, \log N)} \\ \text{mk}\{k\} \leftarrow \text{PuncMK}(\text{mk}, k) \\ \text{td} \leftarrow \text{Trapdoor}(\text{mk}, k) \\ \text{td}\{y\} \leftarrow \text{PuncTD}(\text{td}, y) \\ \hat{x} \leftarrow \mathbf{A}(\text{crs}, y, \text{mk}\{k\}, k, \text{td}\{y\}) \end{array} \right].$$

This is the fixed-output two-level punctured security experiment of Definition 6.1. By assumption, $p_y \leq \epsilon(\lambda, N)$ for every y . In the random experiment, y^* is uniform and independent of setup and the sampled key. Conditional on $y^* = y$, the entire view, including both punctured keys, has exactly the distribution defining p_y . Hence,

$$\delta_{\text{rand}} = \frac{1}{N} \sum_{y \in [N]} p_y \leq \epsilon(\lambda, N).$$

Combining the inequalities gives

$$p_{\text{src}} \leq \epsilon_{\text{src-pr}}(\lambda, N, \epsilon_{\mathcal{D}}) + \epsilon(\lambda, N).$$

The reduction uses only $\mathbf{A}$ and public evaluation, and needs neither the master key, the full trapdoor, nor an efficient source sampler. Its overhead is polynomial. For the concrete instantiation of Construction 6.8, Theorems 6.5 and 6.6 give the bound $O(\epsilon_{\mathcal{D}}(N)) + \text{negl}(\lambda)$. $\square$

7 Deterministic identity-based encryption

We consider an adaptive-identity indistinguishability notion for deterministic identity-based encryption. The challenge message is sampled from a hard weak-source distribution, and the adversary receives the secret key for the challenge identity punctured at the challenge ciphertext. This punctured key still decrypts every other ciphertext for that identity, giving the adversary CCA-style decryption access. Security requires that the challenge ciphertext be indistinguishable from a uniformly random ciphertext, even under this auxiliary information. For related deterministic IBE notions, see Xie, Xue, and Zhang [XXZ12]; for standard CCA-secure DIBE, see Huang et al. [HY-ZLXY19].

7.1 Definition

Definition 7.1 (Puncturable deterministic identity-based encryption). *Let $N = N(\lambda)$ be an efficiently computable domain-size function that is exponential in λ , and let $[N]$ be both the message space and the ciphertext space. Let $\mathcal{I} = \{\mathcal{I}_\lambda\}_{\lambda \in \mathbb{N}}$ be an identity-space ensemble. Let $\ell_{\text{RO}} : \mathbb{N} \rightarrow \mathbb{N}$ be a polynomially bounded function, and let $\mathcal{H}_\lambda$ denote the distribution of a random oracle $f : \{0, 1\}^* \rightarrow \{0, 1\}^{\ell_{\text{RO}}(\lambda)}$. A puncturable deterministic identity-based encryption scheme for $\mathcal{I}$, $\text{D-IBE} = (\text{IBE.Setup}^f, \text{IBE.KGen}^f, \text{IBE.Enc}^f, \text{IBE.Dec}^f, \text{IBE.Puncture}^f)$ consists of the following five PPT oracle algorithms, all of which share access to the same random oracle f .*

- **Setup.** *The randomized algorithm IBE.Setup^f takes as input 1^λ and a domain size N , and outputs public parameters PP and a master secret key MSK .*
- **Key generation.** *The randomized algorithm IBE.KGen^f takes as input PP , MSK , and an identity $\text{id} \in \mathcal{I}_\lambda$, and outputs a secret key sk_{id} .*
- **Encryption.** *The deterministic algorithm IBE.Enc^f takes as input PP , an identity $\text{id} \in \mathcal{I}_\lambda$, and a message $m \in [N]$, and outputs a ciphertext c .*
- **Decryption.** *The deterministic algorithm IBE.Dec^f takes as input PP , an identity $\text{id} \in \mathcal{I}_\lambda$, a secret key for that identity (possibly punctured), and a ciphertext c , and outputs a message or $\perp$.*
- **Puncturing.** *The algorithm IBE.Puncture^f takes as input PP , an identity $\text{id} \in \mathcal{I}_\lambda$, its secret key sk_{id} , and a ciphertext c^* , and outputs the punctured key $\text{sk}\{c^*\}_{\text{id}}$.*

The scheme satisfies the following properties.

- **Correctness.** *For every $\lambda \in \mathbb{N}$, identity $\text{id} \in \mathcal{I}_\lambda$, and message $m \in [N]$,*

$$\Pr \left[\text{IBE.Dec}^f(\text{PP}, \text{id}, \text{sk}_{\text{id}}, \text{IBE.Enc}^f(\text{PP}, \text{id}, m)) = m \mid \begin{array}{l} f \leftarrow \mathcal{H}_\lambda \\ (\text{PP}, \text{MSK}) \leftarrow \text{IBE.Setup}^f(1^\lambda, N) \\ \text{sk}_{\text{id}} \leftarrow \text{IBE.KGen}^f(\text{PP}, \text{MSK}, \text{id}) \end{array} \right] = 1 .$$

- **Punctured correctness.** *For every $\lambda \in \mathbb{N}$, identity $\text{id} \in \mathcal{I}_\lambda$, and ciphertexts $c, c^* \in \mathcal{C}$ such that $c \neq c^*$,*

$$\Pr \left[\begin{array}{l} \text{IBE.Dec}^f(\text{PP}, \text{id}, \text{sk}\{c^*\}_{\text{id}}, c) = \text{IBE.Dec}^f(\text{PP}, \text{id}, \text{sk}_{\text{id}}, c) \\ \wedge \text{IBE.Dec}^f(\text{PP}, \text{id}, \text{sk}\{c^*\}_{\text{id}}, c^*) = \perp \end{array} \mid \begin{array}{l} f \leftarrow \mathcal{H}_\lambda \\ (\text{PP}, \text{MSK}) \leftarrow \text{IBE.Setup}^f(1^\lambda, N) \\ \text{sk}_{\text{id}} \leftarrow \text{IBE.KGen}^f(\text{PP}, \text{MSK}, \text{id}) \\ \text{sk}\{c^*\}_{\text{id}} \leftarrow \text{IBE.Puncture}^f(\text{PP}, \text{id}, \text{sk}_{\text{id}}, c^*) \end{array} \right] = 1 .$$

- Adaptive weak-source pseudorandomness. Let $\epsilon : \mathbb{N} \rightarrow [0, 1]$ and $q : \mathbb{N} \rightarrow \mathbb{N}$ be functions, and let $\mathcal{M} = \{\mathcal{M}_N\}_{N \in \mathbb{N}}$ be an ensemble of distributions over $[N]$ that is (ϵ, q) -hard according to Definition 3.4. We say that the scheme is adaptively pseudorandom with respect to $\mathcal{M}$ if, for every PPT adversary $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2)$,

$$\begin{aligned}
& \Pr \left[\mathbf{A}_2^{\text{IBE.KGen}^f(\text{PP}, \text{MSK}, \cdot), f}(\text{PP}, \text{st}, c^*, \text{sk}\{c^*\}_{\text{id}^*}) = 1 \right] \\
& - \Pr \left[\mathbf{A}_2^{\text{IBE.KGen}^f(\text{PP}, \text{MSK}, \cdot), f}(\text{PP}, \text{st}, c^*, \text{sk}\{c^*\}_{\text{id}^*}) = 1 \right] \\
& \leq O(\epsilon(N)) + \text{negl}(\lambda) .
\end{aligned}$$

$$\begin{aligned}
& f \leftarrow \mathcal{H}_\lambda \\
& (\text{PP}, \text{MSK}) \leftarrow \text{IBE.Setup}^f(1^\lambda, N) \\
& (\text{id}^*, \text{st}) \leftarrow \mathbf{A}_1^{\text{IBE.KGen}^f(\text{PP}, \text{MSK}, \cdot), f}(\text{PP}) \\
& m \leftarrow \mathcal{M}_N \\
& c^* \leftarrow \text{IBE.Enc}^f(\text{PP}, \text{id}^*, m) \\
& \text{sk}_{\text{id}^*} \leftarrow \text{IBE.KGen}^f(\text{PP}, \text{MSK}, \text{id}^*) \\
& \text{sk}\{c^*\}_{\text{id}^*} \leftarrow \text{IBE.Puncture}^f(\text{PP}, \text{id}^*, \text{sk}_{\text{id}^*}, c^*)
\end{aligned}$$

$$\begin{aligned}
& f \leftarrow \mathcal{H}_\lambda \\
& (\text{PP}, \text{MSK}) \leftarrow \text{IBE.Setup}^f(1^\lambda, N) \\
& (\text{id}^*, \text{st}) \leftarrow \mathbf{A}_1^{\text{IBE.KGen}^f(\text{PP}, \text{MSK}, \cdot), f}(\text{PP}) \\
& c^* \leftarrow [N] \\
& \text{sk}_{\text{id}^*} \leftarrow \text{IBE.KGen}^f(\text{PP}, \text{MSK}, \text{id}^*) \\
& \text{sk}\{c^*\}_{\text{id}^*} \leftarrow \text{IBE.Puncture}^f(\text{PP}, \text{id}^*, \text{sk}_{\text{id}^*}, c^*)
\end{aligned}$$

where neither $\mathbf{A}_1$ nor $\mathbf{A}_2$ may query its key-generation oracle on id^* . Random-oracle queries are unrestricted. The punctured key decrypts every challenge-identity ciphertext other than c^* and rejects c^* .

Theorem 7.2 (Adaptive weak-source pseudorandomness). Fix $q(\cdot) = \text{polylog}(\cdot)$. Let $N = N(\lambda)$ be any efficiently computable domain-size function that is exponential in λ . Let $\epsilon_{\mathcal{M}} : \mathbb{N} \rightarrow [0, 1]$ be a function and let $\mathcal{M} = \{\mathcal{M}_N\}_{N \in \mathbb{N}}$ be an $(\epsilon_{\mathcal{M}}, q)$ -hard ensemble of distributions over $[N]$. Suppose that $\epsilon_{\mathcal{M}}(N) = \text{negl}(\lambda)$, $2^{-\ell(\lambda, \log N)} = \text{negl}(\lambda)$, and the dense-key puncturable trapdoor permutation used in Construction 7.3 satisfies the correctness and consistency properties of Definition 6.1 and weak-source pseudorandomness of Definition 6.3 for $\mathcal{M}$, with security error

$$\epsilon_{\text{src-pr}}(\lambda, N, \epsilon_{\mathcal{M}}) \leq O(\epsilon_{\mathcal{M}}(N)) + \text{negl}(\lambda) .$$

Then Construction 7.3 satisfies adaptive weak-source pseudorandomness as defined in Definition 7.1 with respect to $\mathcal{M}$. In particular, under the source and key-length conditions above, the scheme can be instantiated assuming a PPRP for general swaps, indistinguishability obfuscation, and an injective length-doubling PRG, all secure against non-uniform polynomial-time adversaries with negligible advantage.

7.2 Construction and proof

Let $N = N(\lambda)$ be an efficiently computable domain-size function, and let

$$(\text{Gen}, \Pi, \Pi^{-1}, \text{Trapdoor}, \text{PuncMK}, \text{PuncTD})$$

be a dense-key puncturable trapdoor permutation as in Definition 6.1. Its message and ciphertext spaces are $[N]$, and its dense keys are strings in $\{0, 1\}^{\ell(\lambda, \log N)}$. Set

$$\ell_{\text{RO}}(\lambda) := \ell(\lambda, \log N) ,$$

so that the random oracle maps every identity to a dense encrypted key

$$K_{\text{enc}} := f(\text{id}) \in \{0, 1\}^{\ell(\lambda, \log N)}.$$

The puncturable DIBE scheme is defined as follows.

Construction 7.3 (Puncturable DIBE in the random-oracle model). The construction is as follows.

- **IBE.Setup^f**($1^\lambda, N$):
 1. Compute $(\text{crs}, \text{mk}) \leftarrow \text{Gen}(1^\lambda, N)$.
 2. Output $\text{PP} := (\text{crs}, N)$ and $\text{MSK} := \text{mk}$.
- **IBE.KGen^f**($\text{PP}, \text{MSK}, \text{id}$):
 1. Parse $\text{PP} = (\text{crs}, N)$ and $\text{MSK} = \text{mk}$.
 2. Compute $K_{\text{enc}} \leftarrow f(\text{id})$.
 3. Compute $\text{td} \leftarrow \text{Trapdoor}(\text{mk}, K_{\text{enc}})$.
 4. Output $\text{sk}_{\text{id}} := (K_{\text{enc}}, \text{td})$.
- **IBE.Enc^f**(PP, id, m):
 1. Parse $\text{PP} = (\text{crs}, N)$.
 2. Compute $K_{\text{enc}} \leftarrow f(\text{id})$.
 3. Compute $c \leftarrow \Pi(\text{crs}, K_{\text{enc}}, m)$.
 4. Output c .
- **IBE.Dec^f**($\text{PP}, \text{id}, \text{sk}_{\text{id}}, c$):
 1. Parse $\text{PP} = (\text{crs}, N)$ and $\text{sk}_{\text{id}} = (K_{\text{enc}}, \text{td})$.
 2. Compute $\widehat{K}_{\text{enc}} \leftarrow f(\text{id})$.
 3. Output $m \leftarrow \Pi^{-1}(\text{td}, c)$.
- **IBE.Puncture^f**($\text{PP}, \text{id}, \text{sk}_{\text{id}}, c^*$):
 1. Parse $\text{PP} = (\text{crs}, N)$ and $\text{sk}_{\text{id}} = (K_{\text{enc}}, \text{td})$.
 2. Compute $\text{td}\{c^*\} \leftarrow \text{PuncTD}(\text{td}, c^*)$.
 3. Output $\text{sk}\{c^*\}_{\text{id}} := (K_{\text{enc}}, \text{td}\{c^*\})$.

For a punctured key $\text{sk}\{c^*\}_{\text{id}} = (K_{\text{enc}}, \text{td}\{c^*\})$, the decryption algorithm outputs $\Pi^{-1}(\text{td}\{c^*\}, c)$. Correctness and punctured correctness follow from the corresponding properties of Definition 6.1.

Proof. Correctness and punctured correctness follow from the corresponding properties in Definition 6.1. Fix a PPT adversary $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2)$. Define its acceptance probabilities in the real and random experiments by

$$\delta_{\text{real}} := \Pr \left[a = 1 \mid \begin{array}{l} f \leftarrow \mathcal{H}_\lambda \\ (\text{PP}, \text{MSK}) \leftarrow \text{IBE.Setup}^f(1^\lambda, N) \\ (\text{id}^*, \text{st}) \leftarrow \mathbf{A}_1^{\text{IBE.KGen}^f(\text{PP}, \text{MSK}, \cdot), f}(\text{PP}) \\ m^* \leftarrow \mathcal{M}_N \\ c^* \leftarrow \text{IBE.Enc}^f(\text{PP}, \text{id}^*, m^*) \\ \text{sk}_{\text{id}^*} \leftarrow \text{IBE.KGen}^f(\text{PP}, \text{MSK}, \text{id}^*) \\ \text{sk}\{c^*\}_{\text{id}^*} \leftarrow \text{IBE.Puncture}^f(\text{PP}, \text{id}^*, \text{sk}_{\text{id}^*}, c^*) \\ a \leftarrow \mathbf{A}_2^{\text{IBE.KGen}^f(\text{PP}, \text{MSK}, \cdot), f}(\text{PP}, \text{st}, c^*, \text{sk}\{c^*\}_{\text{id}^*}) \end{array} \right], \quad (10)$$

and

$$\delta_{\text{rand}} := \Pr \left[a = 1 \mid \begin{array}{l} f \leftarrow \mathcal{H}_\lambda \\ (\text{PP}, \text{MSK}) \leftarrow \text{IBE.Setup}^f(1^\lambda, N) \\ (\text{id}^*, \text{st}) \leftarrow \mathbf{A}_1^{\text{IBE.KGen}^f(\text{PP}, \text{MSK}, \cdot), f}(\text{PP}) \\ c^* \leftarrow [N] \\ \text{sk}_{\text{id}^*} \leftarrow \text{IBE.KGen}^f(\text{PP}, \text{MSK}, \text{id}^*) \\ \text{sk}\{c^*\}_{\text{id}^*} \leftarrow \text{IBE.Puncture}^f(\text{PP}, \text{id}^*, \text{sk}_{\text{id}^*}, c^*) \\ a \leftarrow \mathbf{A}_2^{\text{IBE.KGen}^f(\text{PP}, \text{MSK}, \cdot), f}(\text{PP}, \text{st}, c^*, \text{sk}\{c^*\}_{\text{id}^*}) \end{array} \right]. \quad (11)$$

The key-generation restrictions in Definition 7.1 apply in both experiments.

We sample the random oracle lazily. Each new input receives an independent uniform value in $\{0, 1\}^{\ell(\lambda, \log N)}$, which is stored for later queries. Thus, if id^* has not been queried before it is selected, then $K_{\text{enc}}^* = f(\text{id}^*)$ is uniform and independent of the preceding view. The definition also permits prior random-oracle queries at id^* . To handle those, we guess its first appearance and embed a uniform dense key at that time.

Let $Q = Q(\lambda)$ be a polynomial bound on the number of distinct inputs appearing in all random-oracle and key-generation queries, across both stages. Count the selected challenge identity as an additional appearance if it has not appeared earlier. There are at most $Q + 1$ first appearances. A key-generation query first looks up the identity's oracle value, creating its entry if necessary.

For $i \in \{0, 1, 2, 3\}$, let δ_i be the probability that Hybrid i outputs 1.

Hybrid 0: Run the real IBE experiment in (10). Let a be the bit output by $\mathbf{A}_2$, and let $P \in [Q + 1]$ be the first-appearance position of id^* , including its challenge lookup. Independently sample $J \leftarrow [Q + 1]$. If $P = J$, then output a . Else, output 0. Formally,

$$\delta_0 := \Pr \left[a = 1 \mid P = J \mid \begin{array}{l} \text{Run (10), obtaining } a, P \\ J \leftarrow [Q + 1] \text{ independently} \end{array} \right].$$

Since J is independent of the honest execution,

$$\delta_0 = \frac{\delta_{\text{real}}}{Q + 1}.$$

Thus, this additional check scales the acceptance probability by $1/(Q + 1)$. We account for this factor at the end of the proof.

Hybrid 1: Here, we use the punctured master key to answer key-generation queries, and use the supplied dense-key challenge at the guessed identity at position J . Our goal is to prove

$$|\delta_1 - \delta_0| \leq \frac{Q}{2^{\ell(\lambda, \log N)}}. \quad (12)$$

We first define δ_1 , then rewrite δ_0 with the same oracle sampling, and finally compare the two experiments.

We construct the following PPT distinguisher $\mathbf{A}'$ for Definition 6.3. On input

$$(\text{crs}, c^*, \text{mk}\{K_{\text{enc}}^*\}, K_{\text{enc}}^*, \text{td}\{c^*\}),$$

it maintains a table T and a counter n . An entry $T[s] = (K_{\text{enc}}, j)$ records the oracle value K_{enc} assigned to s and its first-appearance position j . Write $T[s]_1 = K_{\text{enc}}$ and $T[s]_2 = j$. The simulated oracles $\hat{f}$ and $\hat{K}$ share this state throughout both adversarial stages. An abort in either oracle terminates the entire execution of $\mathbf{A}'$ with output 0.

$\widehat{f}(s)$:

1. If T has an entry s , then output $T[s]_1$.
2. Set $n \leftarrow n + 1$.
3. If $n = J$, then set $K_{\text{enc}} \leftarrow K_{\text{enc}}^*$.
4. Else,
 1. Sample $K_{\text{enc}} \leftarrow \{0, 1\}^{\ell(\lambda, \log N)}$.
 2. If $K_{\text{enc}} = K_{\text{enc}}^*$, then abort and output 0.
5. Set $T[s] \leftarrow (K_{\text{enc}}, n)$ and output K_{enc} .

$\widehat{\mathcal{K}}(\text{id})$:

1. Compute $K_{\text{enc}} \leftarrow \widehat{f}(\text{id})$.
2. If $K_{\text{enc}} = K_{\text{enc}}^*$, then abort and output 0.
3. Else, compute $\text{td} \leftarrow \text{Trapdoor}(\text{mk}\{K_{\text{enc}}^*\}, K_{\text{enc}})$ and output $(K_{\text{enc}}, \text{td})$.

If a randomly sampled oracle value equals K_{enc}^* , a later key-generation query for that identity would require the full trapdoor at K_{enc}^* , which the reduction cannot derive from $\text{mk}\{K_{\text{enc}}^*\}$. For simplicity, $\widehat{f}$ aborts immediately upon detecting this collision. This check applies only when $n \neq J$. Assigning K_{enc}^* at position J does not itself cause an abort.

The execution of $\mathbf{A}'(\text{crs}, c^*, \text{mk}\{K_{\text{enc}}^*\}, K_{\text{enc}}^*, \text{td}\{c^*\})$ is

1. Sample $J \leftarrow [Q + 1]$.
2. Initialize $T \leftarrow \emptyset$ and $n \leftarrow 0$.
3. Set $\text{PP} \leftarrow (\text{crs}, N)$.
4. Emulate $(\text{id}^*, \text{st}) \leftarrow \mathbf{A}_1^{\widehat{\mathcal{K}}, \widehat{f}}(\text{PP})$.
5. Compute $K_{\text{enc}} \leftarrow \widehat{f}(\text{id}^*)$.
6. If $T[\text{id}^*]_2 \neq J$, then output 0.
7. Else, emulate $a \leftarrow \mathbf{A}_2^{\widehat{\mathcal{K}}, \widehat{f}}(\text{PP}, \text{st}, c^*, (K_{\text{enc}}^*, \text{td}\{c^*\}))$.
8. Output a .

The full experiment defining δ_1 is

$$\delta_1 := \Pr \left[\mathbf{A}'(\text{crs}, c^*, \text{mk}\{K_{\text{enc}}^*\}, K_{\text{enc}}^*, \text{td}\{c^*\}) = 1 \mid \begin{array}{l} m^* \leftarrow \mathcal{M}_N \\ (\text{crs}, \text{mk}) \leftarrow \text{Gen}(1^\lambda, N) \\ K_{\text{enc}}^* \leftarrow \{0, 1\}^{\ell(\lambda, \log N)} \\ c^* \leftarrow \Pi(\text{crs}, K_{\text{enc}}^*, m^*) \\ \text{mk}\{K_{\text{enc}}^*\} \leftarrow \text{PuncMK}(\text{mk}, K_{\text{enc}}^*) \\ \text{td} \leftarrow \text{Trapdoor}(\text{mk}, K_{\text{enc}}^*) \\ \text{td}\{c^*\} \leftarrow \text{PuncTD}(\text{td}, c^*) \end{array} \right]. \quad (13)$$

The supplied ciphertext and punctured trapdoor are not used before the challenge step. Since the message ensemble is fixed independently of the setup and adversary, sampling the hidden message before running $\mathbf{A}_1$ does not change the experiment. Its table has at most $Q + 1$ entries, and each simulated query uses polynomial time, so $\mathbf{A}'$ is a PPT distinguisher.

We rewrite δ_0 so that the oracle uses the supplied key K_{enc}^* at position J , just as in Hybrid 1. Both expressions below describe Hybrid 0. In Step 3, we use only the last expression, with oracle f^* . For the two equivalent experiments below, define the oracles separately as follows. When an input s first appears, let j be its first-appearance position in that execution. Each oracle stores its

reply and reuses it on repeated inputs.

$$f \leftarrow \mathcal{H}_\lambda,$$

$$f^*(s) := \begin{cases} K_{\text{enc}}^* & j = J, \\ K_{\text{enc},j} & j \neq J. \end{cases}$$

The oracle f is sampled afresh in the first experiment. The values defining f^* are sampled in the second experiment. Unused values are ignored. Running (10) with f or f^* means using that oracle throughout, instead of sampling another oracle. Each probability includes the indicated oracle sampling. The keys and J are sampled independently of the setup, message, and adversarial coins. We have the following exact equalities.

$$\begin{aligned} \delta_0 &= \Pr \left[a_0 = 1 \left| \begin{array}{l} \text{Run (10) with } f, \text{ obtaining } a, P \\ J \leftarrow [Q+1] \\ a_0 \leftarrow a \cdot \mathbf{1}_{P=J} \end{array} \right. \right] \\ &= \Pr \left[a_0 = 1 \left| \begin{array}{l} J \leftarrow [Q+1] \\ K_{\text{enc},j} \leftarrow \{0,1\}^{\ell(\lambda, \log N)} \quad (j \in [Q+1]) \\ K_{\text{enc}}^* \leftarrow \{0,1\}^{\ell(\lambda, \log N)} \\ \text{Run (10) with } f^*, \text{ obtaining } a, P \\ a_0 \leftarrow a \cdot \mathbf{1}_{P=J} \end{array} \right. \right]. \end{aligned} \quad (14)$$

For every fixed J , the oracle f^* assigns independent uniform replies, including K_{enc}^* at position J . Thus, it has the same distribution as f , and sampling J before the execution preserves its independence from the honest view.

Couple the last experiment in (14) with (13), using the same setup, message, J, K_{enc}^* , presampled values $K_{\text{enc},j}$, and adversarial and puncturing coins. Hybrid 1 uses $K_{\text{enc},j}$ for its fresh draw at position $j \neq J$. Let a_0 and a_1 be the respective outputs, including 0 on any abort in Hybrid 1. The honest execution continues after such an abort. For its adversarial output a and challenge position P ,

$$a_0 = a \cdot \mathbf{1}_{P=J}, \quad \Pr[a_0 = 1] = \delta_0, \quad \Pr[a_1 = 1] = \delta_1.$$

All probabilities below use this coupling. Let Coll be the event that $\hat{f}$ draws $K_{\text{enc}} = K_{\text{enc}}^*$ at a fresh position $j \neq J$ in Hybrid 1. We will show that the outputs agree outside Coll . Then, we bound Coll probability to obtain

$$|\delta_1 - \delta_0| \leq \Pr[a_1 \neq a_0] \leq \Pr[\text{Coll}] \leq \frac{Q}{2^{\ell(\lambda, \log N)}}.$$

On $\neg \text{Coll}$,

- (i) If $P = J$, then $f^*(\text{id}^*) = K_{\text{enc}}^*$, and key-generation queries at id^* are forbidden. All other answers agree by trapdoor consistency in Definition 6.1. The challenge ciphertext and punctured trapdoor also agree under the shared coins, so $a_1 = a_0 = a$.
- (ii) If $P \neq J$, then $a_0 = 0$. Hybrid 1 either aborts earlier or outputs 0 at the check $T[\text{id}^*]_2 \neq J$, so $a_1 = a_0 = 0$.

Thus, $a_1 \neq a_0 \subseteq \text{Coll}$. Therefore, by union bound,

$$\begin{aligned} \Pr[\text{Coll}] &\leq \Pr[\exists j \in [Q+1] \setminus \{J\} \text{ s.t. } K_{\text{enc},j} = K_{\text{enc}}^*] \\ &\leq \sum_{\exists j \in [Q+1] \setminus \{J\}} \Pr[K_{\text{enc},j} = K_{\text{enc}}^*] \\ &= \frac{Q}{2^{\ell(\lambda, \log N)}}. \end{aligned}$$

Consequently,

$$|\delta_1 - \delta_0| \leq \Pr[a_1 \neq a_0] \leq \Pr[\text{Coll}] \leq \frac{Q}{2^{\ell(\lambda, \log N)}},$$

proving (12).

Hybrid 2: In this hybrid, replace the ciphertext by a uniform $c^* \leftarrow [N]$, and puncture the trapdoor at this new ciphertext. The distinguisher $\mathbf{A}'$ is unchanged. The full experiment is

$$\delta_2 := \Pr \left[\mathbf{A}'(\text{crs}, c^*, \text{mk}\{K_{\text{enc}}^*\}, K_{\text{enc}}^*, \text{td}\{c^*\}) = 1 \mid \begin{array}{l} (\text{crs}, \text{mk}) \leftarrow \text{Gen}(1^\lambda, N) \\ K_{\text{enc}}^* \leftarrow \{0, 1\}^{\ell(\lambda, \log N)} \\ \text{mk}\{K_{\text{enc}}^*\} \leftarrow \text{PuncMK}(\text{mk}, K_{\text{enc}}^*) \\ \text{td} \leftarrow \text{Trapdoor}(\text{mk}, K_{\text{enc}}^*) \\ \text{td}\{c^*\} \leftarrow \text{PuncTD}(\text{td}, c^*) \end{array} \right]. \quad (15)$$

By the assumed weak-source pseudorandomness,

$$|\delta_2 - \delta_1| \leq \epsilon_{\text{src-pr}}(\lambda, N, \epsilon_{\mathcal{M}}). \quad (16)$$

Hybrid 3: Here, return to answering all queries honestly, without collision aborts, while retaining the uniform ciphertext and the check $P = J$. Let a and P be the adversary's bit and the challenge identity's first-appearance position in (11). Independently sample $J \leftarrow [Q+1]$. If $P = J$, then output a . Else, output 0. Formally,

$$\delta_3 := \Pr \left[a = 1 \wedge P = J \mid \begin{array}{l} \text{Run (11), obtaining } a, P \\ J \leftarrow [Q+1] \text{ independently} \end{array} \right].$$

As in Hybrid 0, independence of J gives

$$\delta_3 = \frac{\delta_{\text{rand}}}{Q+1}.$$

Use the same coupling as between Hybrids 0 and 1, now with a uniform ciphertext. Let a_2 and a_3 be the outputs of Hybrids 2 and 3, respectively, and let Coll be the collision-abort event in Hybrid 2. On $\neg \text{Coll}$, if $P = J$, then the views and outputs agree. If $P \neq J$, then both hybrids output 0. Thus, under this coupling,

$$\begin{aligned} \{a_2 \neq a_3\} &\subseteq \text{Coll}, \\ |\delta_3 - \delta_2| &\leq \Pr[a_2 \neq a_3] \leq \Pr[\text{Coll}] \leq \frac{Q}{2^{\ell(\lambda, \log N)}}. \end{aligned} \quad (17)$$

The collision bound is unchanged because it depends only on the independent oracle values, not on the ciphertext distribution.

Distinguishing advantage. We bound the distinguishing advantage

$$\Delta := |\delta_{\text{real}} - \delta_{\text{rand}}|,$$

where the probabilities are defined in (10) and (11). By the identities for δ_0 and δ_3 and the triangle inequality,

$$\begin{aligned} \Delta &= (Q+1)|\delta_0 - \delta_3| \\ &\leq (Q+1)(|\delta_0 - \delta_1| + |\delta_1 - \delta_2| + |\delta_2 - \delta_3|) \\ &\leq (Q+1) \left(\epsilon_{\text{src-pr}}(\lambda, N, \epsilon_{\mathcal{M}}) + \frac{2Q}{2^{\ell(\lambda, \log N)}} \right) \\ &\leq (Q+1) \left(O(\epsilon_{\mathcal{M}}(N)) + \text{negl}(\lambda) + \frac{2Q}{2^{\ell(\lambda, \log N)}} \right) \\ &= \text{negl}(\lambda). \end{aligned}$$

The last equality uses the theorem's assumptions that $\epsilon_{\mathcal{M}}(N)$ and $2^{-\ell(\lambda, \log N)}$ are negligible, and that Q is polynomially bounded. This implies the bound $O(\epsilon_{\mathcal{M}}(N)) + \text{negl}(\lambda)$ required by Definition 7.1. Every reduction above runs in polynomial time, and the factor $Q+1$ is polynomially bounded. Thus ordinary negligible security error suffices; the adaptive choice of identity does not require complexity leveraging. For the claimed instantiation, apply Theorem 6.6, whose assumptions are a secure general-swap PPRP, secure iO, and a secure injective length-doubling PRG. Its error bound has no multiplicative N loss, so standard security of these primitives gives the negligible error used above. $\square$

8 Identity-based unique signatures

Identity-based signatures with computational outsider uniqueness were previously studied by Hovd and Stam [HS20]. We construct an identity-based signature with adaptive unforgeability and perfect uniqueness under honestly generated public parameters. Uniqueness is required for each fixed identity–message pair and a fixed random oracle.

8.1 Definition

Definition 8.1 (Identity-based unique signatures). *Let $N = N(\lambda)$ be an efficiently computable domain-size function that is exponential in λ , and let $[N]$ be the signature space. We identify each $i \in [N]$ with the $\lceil \log_2 N \rceil$ -bit binary representation of $i - 1$, including leading zeros, and use elements and their representations interchangeably. All other bit strings are rejected as signatures. Let $\ell : \mathbb{N} \rightarrow \mathbb{N}$ be a polynomially bounded function. Let $\mathbf{U}_\lambda$ denote the distribution of a random oracle*

$$\mathcal{H} : \{0, 1\}^* \longrightarrow \{0, 1\}^{\ell(\lambda)} \times [N].$$

An identity-based unique signature scheme

$$\text{IBS} = (\text{IBS.Setup}^\mathcal{H}, \text{IBS.KGen}^\mathcal{H}, \text{IBS.Sign}^\mathcal{H}, \text{IBS.Verify}^\mathcal{H})$$

consists of the following four PPT oracle algorithms, all of which share access to the same random oracle $\mathcal{H}$.

- **Setup.** *The randomized algorithm $\text{IBS.Setup}^\mathcal{H}$ takes as input 1^λ and a domain size N and outputs public parameters PP and a master secret key MSK .*
- **Key generation.** *The randomized algorithm $\text{IBS.KGen}^\mathcal{H}$ takes as input PP , MSK , and an identity $\text{id} \in \{0, 1\}^*$, and outputs a signing key sk_{id} .*
- **Signing.** *The randomized algorithm $\text{IBS.Sign}^\mathcal{H}$ takes as input PP , an identity id , its signing key sk_{id} , and a message $m \in \{0, 1\}^*$, and outputs a signature $\sigma \in [N]$.*
- **Verification.** *The deterministic algorithm $\text{IBS.Verify}^\mathcal{H}$ takes as input PP , an identity id , a message m , and a signature σ , and outputs a bit.*

The scheme satisfies the following properties for every fixed choice of the oracle function $\mathcal{H}$ and every (PP, MSK) in the support of honest setup:

- **Perfect correctness.** *For every identity $\text{id} \in \{0, 1\}^*$ and message $m \in \{0, 1\}^*$,*

$$\Pr \left[\text{IBS.Verify}^\mathcal{H}(\text{PP}, \text{id}, m, \sigma) = 1 \mid \begin{array}{l} \text{sk}_{\text{id}} \leftarrow \text{IBS.KGen}^\mathcal{H}(\text{PP}, \text{MSK}, \text{id}) \\ \sigma \leftarrow \text{IBS.Sign}^\mathcal{H}(\text{PP}, \text{id}, \text{sk}_{\text{id}}, m) \end{array} \right] = 1 .$$

- **Perfect uniqueness.** *For every identity id , message m , and signature strings σ_0, σ_1 ,*

$$\text{IBS.Verify}^\mathcal{H}(\text{PP}, \text{id}, m, \sigma_0) = \text{IBS.Verify}^\mathcal{H}(\text{PP}, \text{id}, m, \sigma_1) = 1 \implies \sigma_0 = \sigma_1.$$

Thus, for each fixed identity–message pair there is exactly one accepted signature.

Definition 8.2 (Adaptive unforgeability). *The scheme is adaptively unforgeable if, for every non-uniform PPT adversary $\mathbf{A}$,*

$$\Pr \left[\begin{array}{l} \text{IBS.Verify}^{\mathcal{H}}(\text{PP}, \text{id}^*, m^*, \sigma^*) = 1 \\ \wedge \text{id}^* \notin \mathcal{E} \\ \wedge (\text{id}^*, m^*) \notin \mathcal{Q} \end{array} \middle| \begin{array}{l} \mathcal{H} \leftarrow \mathbf{U}_{\lambda} \\ (\text{PP}, \text{MSK}) \leftarrow \text{IBS.Setup}^{\mathcal{H}}(1^{\lambda}, N) \\ \mathcal{E} \leftarrow \emptyset, \quad \mathcal{Q} \leftarrow \emptyset, \quad \mathcal{K} \leftarrow \emptyset \\ (\text{id}^*, m^*, \sigma^*) \leftarrow \mathbf{A}^{\text{Extract, Sign}, \mathcal{H}}(\text{PP}) \end{array} \right] \leq \text{negl}(\lambda) .$$

Here $\mathcal{E}$ records extracted identities, $\mathcal{Q}$ records signed identity–message pairs, and $\mathcal{K}$ is a shared table of signing keys. Whenever either oracle first needs a signing key for id , it samples $\mathcal{K}[\text{id}] \leftarrow \text{IBS.KGen}^{\mathcal{H}}(\text{PP}, \text{MSK}, \text{id})$. The oracles are defined as follows:

- Extraction oracle **Extract**. On input id , add id to $\mathcal{E}$ and return $\mathcal{K}[\text{id}]$.
- Signing oracle **Sign**. On input (id, m) , add (id, m) to $\mathcal{Q}$ and return $\sigma \leftarrow \text{IBS.Sign}^{\mathcal{H}}(\text{PP}, \text{id}, \mathcal{K}[\text{id}], m)$.

8.2 Construction and proof

Let $N = N(\lambda)$ be an efficiently computable domain-size function that is exponential in λ , and let

$$(\text{Gen}, \Pi, \Pi^{-1}, \text{Trapdoor}, \text{PuncMK}, \text{PuncTD})$$

be a dense-key puncturable trapdoor permutation as in Definition 6.1. Its message and ciphertext spaces are $[N]$, and its dense keys are strings in $\{0, 1\}^{\ell(\lambda)}$, where ℓ denotes the key-length function for the fixed domain-size function $N(\lambda)$. Using a fixed efficiently computable and decodable injective encoding of tuples, define

$$\begin{aligned} f(\text{id}) &:= \pi_1(\mathcal{H}(\langle 0, \text{id} \rangle)), \\ h(\text{id}, m) &:= \pi_2(\mathcal{H}(\langle 1, \text{id}, m \rangle)) \quad , \end{aligned}$$

where π_1 and π_2 are the projections onto the first and second coordinates. Thus f and h are independent random functions mapping identities to dense keys and identity–message pairs to $[N]$, respectively. The identity-based unique signature scheme is defined as follows.

Construction 8.3 (Identity-based unique signatures in the random-oracle model). The construction is as follows.

- $\text{IBS.Setup}^{\mathcal{H}}(1^{\lambda}, N)$:
 1. Compute $(\text{crs}, \text{mk}) \leftarrow \text{Gen}(1^{\lambda}, N)$.
 2. Output $\text{PP} := (\text{crs}, N)$ and $\text{MSK} := \text{mk}$.
- $\text{IBS.KGen}^{\mathcal{H}}(\text{PP}, \text{MSK}, \text{id})$:
 1. Parse $\text{PP} = (\text{crs}, N)$ and $\text{MSK} = \text{mk}$.
 2. Compute $K_{\text{enc}} \leftarrow f(\text{id})$.
 3. Compute $\text{td} \leftarrow \text{Trapdoor}(\text{mk}, K_{\text{enc}})$.
 4. Output $\text{sk}_{\text{id}} := (K_{\text{enc}}, \text{td})$.
- $\text{IBS.Sign}^{\mathcal{H}}(\text{PP}, \text{id}, \text{sk}_{\text{id}}, m)$:

1. Parse $PP = (\text{crs}, N)$ and $\text{sk}_{\text{id}} = (K_{\text{enc}}, \text{td})$.
 2. Compute $y \leftarrow h(\text{id}, m)$.
 3. Compute $\sigma \leftarrow \Pi^{-1}(\text{td}, y)$.
 4. Output σ .
- $\text{IBS.Verify}^{\mathcal{H}}(PP, \text{id}, m, \sigma)$:
 1. Parse $PP = (\text{crs}, N)$.
 2. Compute $K_{\text{enc}} \leftarrow f(\text{id})$ and $y \leftarrow h(\text{id}, m)$.
 3. Output 1 if $\Pi(\text{crs}, K_{\text{enc}}, \sigma) = y$, and 0 otherwise.

Perfect correctness follows from trapdoor correctness in Definition 6.1. Perfect uniqueness follows from full key-domain correctness: for every fixed $\mathcal{H}$ and honestly generated crs , $\Pi(\text{crs}, f(\text{id}), \cdot)$ is a permutation of $[N]$, so $h(\text{id}, m)$ has exactly one preimage.

Theorem 8.4 (Adaptive unforgeability). *Let $N = N(\lambda)$ be an efficiently computable domain-size function that is exponential in λ . Suppose that the dense-key puncturable trapdoor permutation used in Construction 8.3 satisfies the correctness properties and two-level punctured security of Definition 6.1 against non-uniform PPT adversaries, with error $\epsilon(\lambda, N) = \text{negl}(\lambda)$, and that $2^{-\ell(\lambda)} = \text{negl}(\lambda)$. Then Construction 8.3 satisfies perfect correctness, perfect uniqueness, and adaptive unforgeability as in Definitions 8.1 and 8.2. In particular, under the key-length condition above, the scheme can be instantiated assuming a PPRP for general swaps, indistinguishability obfuscation, and an injective length-doubling PRG, all secure against non-uniform polynomial-time adversaries with negligible advantage.*

Proof. Perfect correctness and perfect uniqueness follow from the argument above. Fix a non-uniform PPT adversary $\mathbf{A}$, and let δ_{real} be its success probability in Definition 8.2.

Guessing the forged identity and message. Sample f and h lazily, storing their values in tables T_f, T_h . Each $h(\text{id}, m)$ lookup first looks up $f(\text{id})$. After the forgery is output, also look up $h(\text{id}^*, m^*)$. Thus both entries exist even if the adversary never queried them. These internal lookups preserve the adversary's view.

Let $Q = Q(\lambda) \geq 1$ be a polynomial bound on the number of entries in either table, including the final lookups. Sample $J_f, J_h \leftarrow [Q]$ independently, guessing which new identity queried to f and which new identity-message pair queried to h will appear in the forgery. Then

$$\Pr \left[\begin{array}{l} \text{IBS.Verify}^{\mathcal{H}}(PP, \text{id}^*, m^*, \sigma^*) = 1 \\ \wedge \text{id}^* \notin \mathcal{E} \\ \wedge (\text{id}^*, m^*) \notin \mathcal{Q} \\ \wedge \text{pos}_{T_f}(\text{id}^*) = J_f \\ \wedge \text{pos}_{T_h}(\text{id}^*, m^*) = J_h \end{array} \middle| \begin{array}{l} \mathcal{H} \leftarrow \mathbf{U}_{\lambda} \\ (PP, \text{MSK}) \leftarrow \text{IBS.Setup}^{\mathcal{H}}(1^{\lambda}, N) \\ T_f, T_h, \mathcal{E}, \mathcal{Q}, \mathcal{K} \leftarrow \emptyset \\ (\text{id}^*, m^*, \sigma^*) \leftarrow \mathbf{A}^{\text{Extract, Sign}, \mathcal{H}}(PP) \\ h(\text{id}^*, m^*) \\ J_f, J_h \leftarrow [Q] \end{array} \right] = \frac{\delta_{\text{real}}}{Q^2}. \quad (18)$$

Here $\text{pos}_T(x)$ is the number of the query that first records x in T , counting only distinct inputs; the $h(\text{id}^*, m^*)$ line performs the final lookup described above. Equivalently, choose J_f, J_h before the execution and independently sample

$$K_{\text{enc}}^* \leftarrow \{0, 1\}^{\ell(\lambda)}, \quad y^* \leftarrow [N].$$

Assign these values to the J_f -th new f entry and the J_h -th new h entry. Sample all other entries independently and uniformly. For every fixed choice of the guesses, this is the same oracle distribution as in the real experiment.

Reduction to two-level punctured security. Define the following non-uniform PPT adversary, with $\mathbf{A}$ and its advice fixed. Its simulated oracles are defined below. An abort in any oracle terminates $\mathbf{A}'$ with output $\perp$.

$\mathbf{A}'(\text{crs}, y^*, \text{mk}\{K_{\text{enc}}^*\}, K_{\text{enc}}^*, \text{td}\{y^*\})$:

1. Sample $J_f, J_h \leftarrow [Q]$ uniformly at random.
2. Set $\text{PP} \leftarrow (\text{crs}, N)$ and $T_f, T_h \leftarrow \emptyset$. Initialize an empty cache for the remaining coordinates of $\widehat{\mathcal{H}}$.
3. Emulate $(\text{id}^*, m^*, \sigma^*) \leftarrow \mathbf{A}^{\widehat{\text{Extract}}, \widehat{\text{Sign}}, \widehat{\mathcal{H}}}(\text{PP})$.
4. Query $\widehat{h}(\text{id}^*, m^*)$. This also queries $\widehat{f}(\text{id}^*)$, creating both entries if they have not appeared before.
5. Output σ^* .

We define the oracles used in the reduction as follows.

$\widehat{f}(\text{id})$:

1. If $\text{id} \in \text{dom}(T_f)$, output $T_f[\text{id}]$.
2. If $|T_f| + 1 = J_f$, set $T_f[\text{id}] \leftarrow K_{\text{enc}}^*$. Otherwise, sample $T_f[\text{id}] \leftarrow \{0, 1\}^{\ell(\lambda)}$ and abort if $T_f[\text{id}] = K_{\text{enc}}^*$.
3. Output $T_f[\text{id}]$.

$\widehat{h}(\text{id}, m)$:

1. Query $\widehat{f}(\text{id})$. If $(\text{id}, m) \in \text{dom}(T_h)$, output $T_h[(\text{id}, m)]$.
2. If $|T_h| + 1 = J_h$, set $T_h[(\text{id}, m)] \leftarrow y^*$. Otherwise, sample $T_h[(\text{id}, m)] \leftarrow [N]$ and abort if $T_h[(\text{id}, m)] = y^*$.
3. Output $T_h[(\text{id}, m)]$.

$\widehat{\text{Extract}}(\text{id})$:

1. Compute $K_{\text{enc}} \leftarrow \widehat{f}(\text{id})$. If $K_{\text{enc}} = K_{\text{enc}}^*$, abort.
2. Output $(K_{\text{enc}}, \text{Trapdoor}(\text{mk}\{K_{\text{enc}}^*\}, K_{\text{enc}}))$.

$\widehat{\text{Sign}}(\text{id}, m)$:

1. Compute $K_{\text{enc}} \leftarrow \widehat{f}(\text{id})$ and $y \leftarrow \widehat{h}(\text{id}, m)$.
2. If $(K_{\text{enc}}, y) = (K_{\text{enc}}^*, y^*)$, abort.
3. Output $\begin{cases} \Pi^{-1}(\text{td}\{y^*\}, y) & K_{\text{enc}} = K_{\text{enc}}^*, \\ \Pi^{-1}(\text{Trapdoor}(\text{mk}\{K_{\text{enc}}^*\}, K_{\text{enc}}), y) & K_{\text{enc}} \neq K_{\text{enc}}^*. \end{cases}$

For a direct query to $\widehat{\mathcal{H}}$, use $\widehat{f}(\text{id})$ as the first coordinate at $\langle 0, \text{id} \rangle$ and $\widehat{h}(\text{id}, m)$ as the second coordinate at $\langle 1, \text{id}, m \rangle$. Sample and cache all other coordinates independently and uniformly, as in the real oracle.

Let δ_1 be the inversion probability of $\mathbf{A}'$.

$$\delta_1 := \Pr \left[\Pi(\text{crs}, K_{\text{enc}}^*, \sigma^*) = y^* \mid \begin{array}{l} y^* \leftarrow [N] \\ (\text{crs}, \text{mk}) \leftarrow \text{Gen}(1^\lambda, N) \\ K_{\text{enc}}^* \leftarrow \{0, 1\}^{\ell(\lambda)} \\ \text{mk}\{K_{\text{enc}}^*\} \leftarrow \text{PuncMK}(\text{mk}, K_{\text{enc}}^*) \\ \text{td} \leftarrow \text{Trapdoor}(\text{mk}, K_{\text{enc}}^*) \\ \text{td}\{y^*\} \leftarrow \text{PuncTD}(\text{td}, y^*) \\ \sigma^* \leftarrow \mathbf{A}'(\text{crs}, y^*, \text{mk}\{K_{\text{enc}}^*\}, K_{\text{enc}}^*, \text{td}\{y^*\}) \end{array} \right]. \quad (19)$$

Applying Definition 6.1 for each fixed y^* and averaging gives

$$\delta_1 \leq \epsilon(\lambda, N). \quad (20)$$

Comparing the experiments. Couple the real experiment with the simulation using the same setup, guesses, designated values, oracle samples, and adversary coins. Let **Coll** be the event that a real f entry other than the J_f -th receives K_{enc}^* , or a real h entry other than the J_h -th receives y^* . Every non-designated entry is sampled uniformly, independently of its designated value and the preceding transcript. Hence

$$\Pr[\text{Coll}] \leq \frac{Q}{2^{\ell(\lambda)}} + \frac{Q}{N}. \quad (21)$$

Suppose the real forgery is valid and fresh, both guesses are correct, and **Coll** does not occur. Then the forged identity is the only queried identity mapped to K_{enc}^* , and the forged pair is the only queried pair mapped to y^* . Freshness excludes extraction of that identity and signing of that pair, so no simulated oracle aborts.

For every other identity, master-key consistency gives the same trapdoor as in the real experiment. Deterministic trapdoor derivation also preserves repeated extraction responses. For signing queries under the forged identity, punctured trapdoor correctness gives the same unique preimage whenever $y \neq y^*$:

$$\Pi^{-1}(\text{td}\{y^*\}, y) = \Pi^{-1}(\text{td}, y).$$

Thus, by induction over queries, the views agree through the final lookups. The reduction outputs the real forgery, satisfying

$$\Pi(\text{crs}, K_{\text{enc}}^*, \sigma^*) = h(\text{id}^*, m^*) = y^*.$$

By (18), the successful real executions with both guesses correct have probability δ_{real}/Q^2 . Only executions in **Coll** can be lost in the simulation, so

$$\begin{aligned} \frac{\delta_{\text{real}}}{Q^2} &\leq \delta_1 + \Pr[\text{Coll}] \\ &\leq \epsilon(\lambda, N) + \frac{Q}{2^{\ell(\lambda)}} + \frac{Q}{N}. \end{aligned} \quad (22)$$

Consequently,

$$\delta_{\text{real}} \leq Q^2 \epsilon(\lambda, N) + \frac{Q^3}{2^{\ell(\lambda)}} + \frac{Q^3}{N} = \text{negl}(\lambda).$$

All reductions preserve the adversary's advice and run in polynomial time. The claimed instantiation follows from the reductions in Theorem 6.5, applied to non-uniform adversaries. $\square$

9 Permutable PRP with a full key domain in the CRS model

In this section, we construct an output-permutable PRP with a full key domain in the CRS model. Under every honestly generated crs , every string of the prescribed key length defines a permutation of the full message domain, together with its inverse. A private permute key allows derivation of a key for either the original permutation or its composition with a chosen output permutation. We require these derived keys to be indistinguishable from uniformly random strings, even when the output permutation is chosen adaptively after seeing the crs . We first define this notion and show that it implies pseudorandomness under a uniformly random key, and then give a construction for the class of general swaps using puncturable PRPs.

9.1 Definition

Definition 9.1 (OP-PRP with a full key domain in the CRS model). *Let $G := \{G_N\}_{N \in \mathbb{N}}$ be a collection where each G_N is a set of permutations over $[N]$. An output-permutable PRP in the CRS model for a collection G is a tuple of algorithms $(\text{Gen}, \Pi, \Pi^{-1}, \text{Permute}, \text{Eval}, \text{Eval}^{-1})$ with the following properties:*

- **Setup.** *The Gen algorithm is a PPT algorithm that takes as input a security parameter 1^λ and a domain size N , where $\lambda, N \in \mathbb{N}$, and outputs a common reference string crs and a private permute key pv .*
- **Efficient permutations.** *The algorithms Π and Π^{-1} are deterministic polynomial-time algorithms that given a key $k \leftarrow \{0, 1\}^\lambda$ computes a permutation over $[N]$ and its inverse, respectively.*
- **Output permute correctness.** *The Permute algorithm is a deterministic polynomial-time algorithm which takes as input a private permute key $(\cdot, \text{pv}) \in \text{Gen}(1^\lambda, N)$, a key $k \in \{0, 1\}^\lambda$, a permutation $\Gamma \in G_N$, and a bit $b \in \{0, 1\}$. It outputs a key $k^{\Gamma, b} \in \{0, 1\}^{\ell(\lambda, \log N)}$, for some fixed polynomial ℓ . For every $\lambda, N \in \mathbb{N}$, $k \in \{0, 1\}^\lambda$, $b \in \{0, 1\}$, $\Gamma \in G_N$ and $x, z \in [N]$:*

$$\begin{aligned} \text{Eval}(\text{crs}, k^{\Gamma, b}, x) &\equiv \begin{cases} \Gamma(\Pi(k, x)) & \text{if } b = 1 \\ \Pi(k, x) & \text{otherwise} \end{cases} \\ \text{Eval}^{-1}(\text{crs}, k^{\Gamma, b}, z) &\equiv \begin{cases} \Pi^{-1}(k, \Gamma^{-1}(z)) & \text{if } b = 1 \\ \Pi^{-1}(k, z) & \text{otherwise} \end{cases} \end{aligned}$$

We refer to $k^{\Gamma, b}$ as a permuted key.

- **Full key-domain correctness.** *For every $\lambda, N \in \mathbb{N}$, $(\text{crs}, \cdot) \in \text{Gen}(1^\lambda, N)$, and for every string $k^* \in \{0, 1\}^{\ell(\lambda, \log N)}$, the function $\text{Eval}(\text{crs}, k^*, \cdot)$ is a permutation over $[N]$, and $\text{Eval}^{-1}(\text{crs}, k^*, \cdot)$ is its inverse.*
- **Pseudorandomness of permuted keys (adaptive security).** *For every PPT adversary $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2)$, and for every $\lambda, N \in \mathbb{N}$, the permuted keys are computationally indistinguishable from uniform*

strings in $\{0, 1\}^{\ell(\lambda, \log N)}$, even when the permutation is chosen adaptively given the crs:

$$\Pr \left[\Gamma \in G_N \wedge c = c' \mid \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ (\Gamma, b, \text{aux}) \leftarrow \mathbf{A}_1(\text{crs}) \\ c \leftarrow \{0, 1\} \\ \mathbf{k} \leftarrow \{0, 1\}^\lambda \\ \mathbf{k}^* \leftarrow \begin{cases} \text{Permute}(\text{pv}, \mathbf{k}, \Gamma, b) & \text{if } c = 0 \\ \{0, 1\}^{\ell(\lambda, \log N)} & \text{if } c = 1 \end{cases} \\ c' \leftarrow \mathbf{A}_2(\text{crs}, \text{aux}, \mathbf{k}^*) \end{array} \right] \leq \frac{1}{2} + \text{negl}(\lambda) .$$

Lemma 9.2. Let $(\text{Gen}, \Pi, \Pi^{-1}, \text{Permute}, \text{Eval}, \text{Eval}^{-1})$ be a secure dense key OP-PRP in the CRS model according to Definition 9.1. Assume that the base permutation Π, Π^{-1} is a secure pseudorandom permutation. That is, for any PPT adversary $\mathbf{A}$ the following holds.

$$\Pr \left[\mathbf{A}^{\Pi(\text{crs}, \mathbf{k}, \cdot), \Pi^{-1}(\text{crs}, \mathbf{k}, \cdot)}(\text{crs}) = 1 \mid \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ \mathbf{k} \leftarrow \{0, 1\}^\lambda \end{array} \right] \\ - \Pr \left[\mathbf{A}^{f, f^{-1}}(\text{crs}) = 1 \mid \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ f \leftarrow U_N \end{array} \right] \leq \text{negl}(\lambda) .$$

Then, Eval is also pseudorandom for key that is sampled uniformly at random. That is,

$$\Pr \left[\mathbf{A}_2^{\text{Eval}(\text{crs}, \mathbf{k}^*, \cdot), \text{Eval}^{-1}(\text{crs}, \mathbf{k}^*, \cdot)}(\text{crs}, \text{aux}) = 1 \mid \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ (\Gamma, b, \text{aux}) \leftarrow \mathbf{A}_1(\text{crs}) \\ \mathbf{k}^* \leftarrow \{0, 1\}^{\ell(\lambda, \log N)} \end{array} \right] \\ - \Pr \left[\mathbf{A}_2^{f, f^{-1}}(\text{crs}, \text{aux}) = 1 \mid \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ (\Gamma, b, \text{aux}) \leftarrow \mathbf{A}_1(\text{crs}) \\ f \leftarrow U_N \end{array} \right] \leq \text{negl}(\lambda) ,$$

Proof. Assume that for every PPT adversary $\mathbf{A}$ the following holds.

$$\Pr \left[\mathbf{A}^{\Pi(\text{crs}, \mathbf{k}, \cdot), \Pi^{-1}(\text{crs}, \mathbf{k}, \cdot)}(\text{crs}) = 1 \mid \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ \mathbf{k} \leftarrow \{0, 1\}^\lambda \end{array} \right] \\ - \Pr \left[\mathbf{A}^{f, f^{-1}}(\text{crs}) = 1 \mid \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ f \leftarrow U_N \end{array} \right] \leq \text{negl}(\lambda) . \quad (23)$$

Fix any PPT adversary $\mathbf{A} := (\mathbf{A}_1, \mathbf{A}_2)$. The pseudorandomness-of-permuted-keys property holds for every choice of $\mathbf{A}_1$, including the choice $b = 0$. Therefore,

$$\Pr \left[\mathbf{A}_2^{\text{Eval}(\text{crs}, \mathbf{k}^*, \cdot), \text{Eval}^{-1}(\text{crs}, \mathbf{k}^*, \cdot)}(\text{crs}, \text{aux}) = 1 \mid \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ (\Gamma, b, \text{aux}) \leftarrow \mathbf{A}_1(\text{crs}) \\ \mathbf{k}^* \leftarrow \{0, 1\}^{\ell(\lambda, \log N)} \end{array} \right] \\ - \Pr \left[\mathbf{A}_2^{\text{Eval}(\text{crs}, \mathbf{k}^*, \cdot), \text{Eval}^{-1}(\text{crs}, \mathbf{k}^*, \cdot)}(\text{crs}, \text{aux}) = 1 \mid \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ (\Gamma, b, \text{aux}) \leftarrow \mathbf{A}_1(\text{crs}) \\ \mathbf{k} \leftarrow \{0, 1\}^\lambda \\ \mathbf{k}^* \leftarrow \text{Permute}(\text{pv}, \mathbf{k}, \Gamma, 0) \end{array} \right] \leq \text{negl}(\lambda) .$$

By the correctness of the permuted key, when k^* is generated by $\text{Permute}(\text{pv}, k, \Gamma, 0)$, the functions $\text{Eval}(\text{crs}, k^*, \cdot)$ and $\Pi(\text{crs}, k, \cdot)$ are functionally equivalent. Therefore, the adversary's view is identically distributed in both experiments. We get that,

$$\begin{aligned} & \Pr \left[\mathbf{A}_2^{\text{Eval}(\text{crs}, k^*, \cdot), \text{Eval}^{-1}(\text{crs}, k^*, \cdot)}(\text{crs}, \text{aux}) = 1 \mid \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ (\Gamma, b, \text{aux}) \leftarrow \mathbf{A}_1(\text{crs}) \\ k \leftarrow \{0, 1\}^\lambda \\ k^* \leftarrow \text{Permute}(\text{pv}, k, \Gamma, 0) \end{array} \right] \\ & - \Pr \left[\mathbf{A}_2^{\Pi(\text{crs}, k, \cdot), \Pi^{-1}(\text{crs}, k, \cdot)}(\text{crs}, \text{aux}) = 1 \mid \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ (\Gamma, b, \text{aux}) \leftarrow \mathbf{A}_1(\text{crs}) \\ k \leftarrow \{0, 1\}^\lambda \end{array} \right] = 0 . \end{aligned}$$

Since $\mathbf{A}_1$ and $\mathbf{A}_2$ in the above experiment can be emulated together on the left side of the probability (the experiment does not use the output of $\mathbf{A}_1$), by Equation 23 (Π is pseudorandom), we get that,

$$\begin{aligned} & \Pr \left[\mathbf{A}_2^{\Pi(\text{crs}, k, \cdot), \Pi^{-1}(\text{crs}, k, \cdot)}(\text{crs}, \text{aux}) = 1 \mid \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ (\Gamma, b, \text{aux}) \leftarrow \mathbf{A}_1(\text{crs}) \\ k \leftarrow \{0, 1\}^\lambda \end{array} \right] \\ & - \Pr \left[\mathbf{A}_2^{f, f^{-1}}(\text{crs}, \text{aux}) = 1 \mid \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ (\Gamma, b, \text{aux}) \leftarrow \mathbf{A}_1(\text{crs}) \\ f \leftarrow U_N \end{array} \right] \leq \text{negl}(\lambda) . \end{aligned}$$

Overall, we get that,

$$\begin{aligned} & \Pr \left[\mathbf{A}_2^{\text{Eval}(\text{crs}, k^*, \cdot), \text{Eval}^{-1}(\text{crs}, k^*, \cdot)}(\text{crs}, \text{aux}) = 1 \mid \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ (\Gamma, b, \text{aux}) \leftarrow \mathbf{A}_1(\text{crs}) \\ k^* \leftarrow \{0, 1\}^{\ell(\lambda, \log N)} \end{array} \right] \\ & - \Pr \left[\mathbf{A}_2^{f, f^{-1}}(\text{crs}, \text{aux}) = 1 \mid \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ (\Gamma, b, \text{aux}) \leftarrow \mathbf{A}_1(\text{crs}) \\ f \leftarrow U_N \end{array} \right] \leq \text{negl}(\lambda) , \end{aligned}$$

which proves the lemma. $\square$

Theorem 9.3. *Assume the existence of indistinguishability obfuscation with sub-exponential security, a secure permutable PRP for the class of general swaps G with sub-exponential security, and a PRG with sub-exponential security. Then, for every efficiently computable domain-size function $N = N(\lambda)$ that is exponential in λ , there exists an OP-PRP in the CRS model for the class of general swaps G with sub-exponential security.*

9.2 Construction and proof

Proof. Let $N \in \mathbb{N}$ be the domain size, and let $\lambda \in \mathbb{N}$ be the security parameter. Let iO be an indistinguishability obfuscation with security $\epsilon_{\text{iO}}(\cdot)$, let $(\text{base}.\Pi, \text{base}.\Pi^{-1}, \text{base}.\text{Permute})$ be a permutable PRP scheme over $[N]$ for the class of general swaps G with security $\epsilon_{\text{PRP}}(\cdot, \cdot)$, and let $\epsilon_{\text{PRG}}(\cdot)$ be the security of the underlying PRG. We assume that all primitives have sub-exponential security. Specifically, there exists some $0 < \epsilon < 1$ such that $\epsilon_{\text{iO}}(\lambda) \leq 2^{-\lambda^\epsilon}$, $\epsilon_{\text{PRP}}(\lambda, N) \leq 2^{-\lambda^\epsilon}$,

and $\epsilon_{\text{PRG}}(N) \leq 2^{-(\log N)^\epsilon}$. By Theorem 4.2, the above primitives implies the existence of a puncturable PRP scheme ($\text{punc.}\Pi$, $\text{punc.}\Pi^{-1}$, $\text{punc.}\text{Permute}$, $\text{punc.}\text{Punc}$, $\text{punc.}\text{Eval}$, $\text{punc.}\text{Eval}^{-1}$) with security

$$\epsilon'_{\text{PRP}}(\lambda, N) = O(1/N + \epsilon_{\text{PRG}}(N) + \epsilon_{\text{iO}}(\lambda) + \epsilon_{\text{PRP}}(\lambda, N)).$$

In this construction, G denotes the space of fixed-length swap descriptions: two $\lceil \log N \rceil$ -bit endpoint fields. Descriptions with an out-of-range endpoint represent the identity. Sampling from G means sampling a uniform description string, not a uniform distinct permutation. Concatenation retains that string. Thus $|G| = 2^{2\lceil \log N \rceil} = O(N^2)$. In the proof, we set an internal security parameter λ' and domain size N' which we define as follows.

$$\begin{aligned}\lambda' &:= (\lceil \log |G| \rceil \cdot \lambda)^{1/\epsilon}, \\ N' &:= |G| \cdot 2^{\lambda'+1}.\end{aligned}$$

Since ϵ is a fixed constant and $|G| = O(N^2)$ (describing swaps over $[N]$), note that $\lambda' = \text{poly}(\lambda, \log N)$, maintaining efficiency.

We define three circuit families parameterized by a master setup key $\mathbf{k}_{\text{master}} \in \{0, 1\}^{\lambda'}$:

$$\begin{aligned}\lambda' &:= (\lceil \log |G| \rceil \cdot \lambda)^{1/\epsilon}, \\ N' &:= |G| \cdot 2^{\lambda'+1}.\end{aligned}$$

Since ϵ is a fixed constant and $|G| = O(N^2)$ (describing swaps over $[N]$), note that $\lambda' = \text{poly}(\lambda, \log N)$, maintaining efficiency.

We define three circuit families parameterized by a master setup key $\mathbf{k}_{\text{master}} \in \{0, 1\}^{\lambda'}$:

$C_{\text{perm}}[\mathbf{k}_{\text{master}}](\Gamma, b, \mathbf{k})$:

1. Output $K_{\text{enc}} \leftarrow \text{punc.}\Pi(\mathbf{k}_{\text{master}}, (\Gamma \parallel b \parallel \mathbf{k}))$.

$C_{\text{eval}}[\mathbf{k}_{\text{master}}](K_{\text{enc}}, x)$:

1. Compute $(\Gamma \parallel b \parallel \mathbf{k}) \leftarrow \text{punc.}\Pi^{-1}(\mathbf{k}_{\text{master}}, K_{\text{enc}})$.
2. If $b = 0$, then output $\text{base.}\Pi(\mathbf{k}, x)$.
3. Else ($b = 1$), output $\Gamma(\text{base.}\Pi(\mathbf{k}, x))$.

$C_{\text{inv}}[\mathbf{k}_{\text{master}}](K_{\text{enc}}, y)$:

1. Compute $(\Gamma \parallel b \parallel \mathbf{k}) \leftarrow \text{punc.}\Pi^{-1}(\mathbf{k}_{\text{master}}, K_{\text{enc}})$.
2. If $b = 0$, then output $\text{base.}\Pi^{-1}(\mathbf{k}, y)$.
3. Else ($b = 1$), output $\text{base.}\Pi^{-1}(\mathbf{k}, \Gamma^{-1}(y))$.

Construction 9.4. The output-permutable PRP scheme in the CRS model is defined as follows.

- $\text{Gen}(1^\lambda, N)$:

1. Set $s := \text{poly}(\lambda, \log N)$, where the polynomial is large enough to upper-bound the size of all modified circuits appearing in the proof.
2. Sample $\mathbf{k}_{\text{master}} \leftarrow \{0, 1\}^{\lambda'}$.
3. Compute
 - (a) $C_{\text{perm}} \leftarrow C_{\text{perm}}[\mathbf{k}_{\text{master}}]$.
 - (b) $\widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{eval}}[\mathbf{k}_{\text{master}}])$.

- (c) $\widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{inv}}[k_{\text{master}}])$.
4. Output (crs, pv) , where $\text{crs} := (\widehat{C_{\text{eval}}}, \widehat{C_{\text{inv}}})$ and $\text{pv} := C_{\text{perm}}$.
- $\Pi(k, x)$:
 1. Output $\text{base}.\Pi(k, x)$.
 - $\Pi^{-1}(k, y)$:
 1. Output $\text{base}.\Pi^{-1}(k, y)$.
 - $\text{Permute}(\text{pv}, k, \Gamma, b)$:
 1. Parse pv as C_{perm} .
 2. Output $K_{\text{enc}} \leftarrow C_{\text{perm}}(\Gamma, b, k)$.
 - $\text{Eval}(\text{crs}, K_{\text{enc}}, x)$:
 1. Parse crs to extract $\widehat{C_{\text{eval}}}$.
 2. Output $y \leftarrow \widehat{C_{\text{eval}}}(K_{\text{enc}}, x)$.
 - $\text{Eval}^{-1}(\text{crs}, K_{\text{enc}}, y)$:
 1. Parse crs to extract $\widehat{C_{\text{inv}}}$.
 2. Output $x \leftarrow \widehat{C_{\text{inv}}}(K_{\text{enc}}, y)$.

We prove that the construction achieves OP-PRP security in the CRS model through a sequence of hybrids. We begin by establishing non-adaptive security for any $\Gamma^* \in G$ and $b^* \in \{0, 1\}$ fixed before the CRS generation. Next, we extend this result to adaptive security, where the adversary selects Γ^* and b^* adaptively given the crs .

Non-adaptive security. We start by a sequence of indistinguishable hybrids. Fix any $\Gamma^* \in G$ and $b^* \in \{0, 1\}$.

Hybrid 0: For a PRP master key $k_{\text{master}} \in \{0, 1\}^{\lambda'}$, let $C_{\text{eval}}[k_{\text{master}}]$ and $C_{\text{inv}}[k_{\text{master}}]$ be the following circuits.

- $C_{\text{eval}}[k_{\text{master}}](K_{\text{enc}}, x)$:
 1. Compute $(\Gamma \parallel b \parallel k) \leftarrow \text{punc}.\Pi^{-1}(k_{\text{master}}, K_{\text{enc}})$.
 2. If $b = 0$, then output $\text{base}.\Pi(k, x)$.
 3. Else ($b = 1$), output $\Gamma(\text{base}.\Pi(k, x))$.
- $C_{\text{inv}}[k_{\text{master}}](K_{\text{enc}}, y)$:
 1. Compute $(\Gamma \parallel b \parallel k) \leftarrow \text{punc}.\Pi^{-1}(k_{\text{master}}, K_{\text{enc}})$.
 2. If $b = 0$, then output $\text{base}.\Pi^{-1}(k, y)$.
 3. Else ($b = 1$), output $\text{base}.\Pi^{-1}(k, \Gamma^{-1}(y))$.

By construction,

$$\left\{ \begin{array}{l|l} (\text{crs}, K_{\text{enc}}^*) & \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ b^* \leftarrow \{0, 1\} \\ k^* \leftarrow \{0, 1\}^\lambda \\ K_{\text{enc}}^* \leftarrow \text{Permute}(\text{pv}, (\Gamma^* \parallel b^* \parallel k^*)) \end{array} \end{array} \right\} \\ \equiv \left\{ \begin{array}{l|l} (\text{crs}, K_{\text{enc}}^*) & \begin{array}{l} k_{\text{master}} \leftarrow \{0, 1\}^{\lambda'} \\ k^* \leftarrow \{0, 1\}^\lambda \\ K_{\text{enc}}^* \leftarrow \text{punc.}\Pi(k_{\text{master}}, (\Gamma^* \parallel b^* \parallel k^*)) \\ \widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{eval}}[k_{\text{master}}]) \\ \widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{inv}}[k_{\text{master}}]) \\ \text{crs} := (\widehat{C_{\text{eval}}}, \widehat{C_{\text{inv}}}) \end{array} \end{array} \right\}.$$

Hybrid 1: For a PRP master key $k_{\text{master}} \in \{0, 1\}^{\lambda'}$, a tuple $M^* := (\Gamma^* \parallel b^* \parallel k^*) \in G \times \{0, 1\} \times \{0, 1\}^{\lambda'}$, and $K_{\text{enc}}^* \in [N']$, we define the circuits $C_{\text{eval}}[k_{\text{master}}, M^*, K_{\text{enc}}^*]$ and $C_{\text{inv}}[k_{\text{master}}, M^*, K_{\text{enc}}^*]$ as follows.

- $C_{\text{eval}}[k_{\text{master}}, M^*, K_{\text{enc}}^*](K_{\text{enc}}, x)$:
 1. If $K_{\text{enc}} = K_{\text{enc}}^*$, then set $(\Gamma \parallel b \parallel k) := M^*$.
 2. Else, compute $(\Gamma \parallel b \parallel k) \leftarrow \text{punc.}\Pi^{-1}(k_{\text{master}}, K_{\text{enc}})$.
 3. If $b = 0$, then output $\text{base.}\Pi(k, x)$.
 4. Else ($b = 1$), output $\Gamma(\text{base.}\Pi(k, x))$.
- $C_{\text{inv}}[k_{\text{master}}, M^*, K_{\text{enc}}^*](K_{\text{enc}}, y)$:
 1. If $K_{\text{enc}} = K_{\text{enc}}^*$, then set $(\Gamma \parallel b \parallel k) := M^*$.
 2. Else, compute $(\Gamma \parallel b \parallel k) \leftarrow \text{punc.}\Pi^{-1}(k_{\text{master}}, K_{\text{enc}})$.
 3. If $b = 0$, then output $\text{base.}\Pi^{-1}(k, x)$.
 4. Else ($b = 1$), output $\text{base.}\Pi^{-1}(k, \Gamma^{-1}(x))$.

Note that for any key tuple M^* , when the encrypted key is honestly computed as $K_{\text{enc}}^* \leftarrow \text{punc.}\Pi(k_{\text{master}}, M^*)$, the circuit $C_{\text{eval}}[k_{\text{master}}, M^*, K_{\text{enc}}^*]$ is simply $C_{\text{eval}}[k_{\text{master}}]$ with hardcoded values. Thus, $C_{\text{eval}}[k_{\text{master}}] \equiv C_{\text{eval}}[k_{\text{master}}, M^*, K_{\text{enc}}^*]$, and similarly $C_{\text{inv}}[k_{\text{master}}] \equiv C_{\text{inv}}[k_{\text{master}}, M^*, K_{\text{enc}}^*]$. The following indistin-

guishability then follows by iO security,

$$\left\{ \begin{array}{l} \text{crs}, K_{\text{enc}}^* \\ \left| \begin{array}{l} k_{\text{master}} \leftarrow \{0, 1\}^{\lambda'} \\ k^* \leftarrow \{0, 1\}^{\lambda} \\ K_{\text{enc}}^* \leftarrow \text{punc.}\Pi(k_{\text{master}}, (\Gamma^* \parallel b^* \parallel k^*)) \\ \widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{eval}}[k_{\text{master}}]) \\ \widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{inv}}[k_{\text{master}}]) \\ \text{crs} := (\widehat{C_{\text{eval}}}, \widehat{C_{\text{inv}}}) \end{array} \right. \end{array} \right\} \\ \approx_{\epsilon_{\text{iO}}(\lambda')} \left\{ \begin{array}{l} \text{crs}, K_{\text{enc}}^* \\ \left| \begin{array}{l} k_{\text{master}} \leftarrow \{0, 1\}^{\lambda'} \\ k^* \leftarrow \{0, 1\}^{\lambda} \\ M^* := (\Gamma^* \parallel b^* \parallel k^*) \\ K_{\text{enc}}^* \leftarrow \text{punc.}\Pi(k_{\text{master}}, M^*) \\ \widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{eval}}[k_{\text{master}}, M^*, K_{\text{enc}}^*]) \\ \widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{inv}}[k_{\text{master}}, M^*, K_{\text{enc}}^*]) \\ \text{crs} := (\widehat{C_{\text{eval}}}, \widehat{C_{\text{inv}}}) \end{array} \right. \end{array} \right\}.$$

Hybrid 2: Let $k_{\text{master}}\{M^*, K_{\text{enc}}^*\}$ be the master key k_{master} punctured at the locations (M^*, K_{enc}^*) . Then, we define $C_{\text{eval}}[k_{\text{master}}\{M^*, K_{\text{enc}}^*\}, M^*, K_{\text{enc}}^*]$ and $C_{\text{inv}}[k_{\text{master}}\{M^*, K_{\text{enc}}^*\}, M^*, K_{\text{enc}}^*]$ as follows.

- $C_{\text{eval}}[k_{\text{master}}\{M^*, K_{\text{enc}}^*\}, M^*, K_{\text{enc}}^*](K_{\text{enc}}, x)$:
 1. If $K_{\text{enc}} = K_{\text{enc}}^*$, then set $(\Gamma \parallel b \parallel k) := M^*$.
 2. Else, compute $(\Gamma \parallel b \parallel k) \leftarrow \text{punc.Eval}^{-1}(k_{\text{master}}\{M^*, K_{\text{enc}}^*\}, K_{\text{enc}})$.
 3. If $b = 0$, then output $\text{base.}\Pi(k, x)$.
 4. Else ($b = 1$), output $\Gamma(\text{base.}\Pi(k, x))$.
- $C_{\text{inv}}[k_{\text{master}}\{M^*, K_{\text{enc}}^*\}, M^*, K_{\text{enc}}^*](K_{\text{enc}}, y)$:
 1. If $K_{\text{enc}} = K_{\text{enc}}^*$, then set $(\Gamma \parallel b \parallel k) := M^*$.
 2. Else, compute $(\Gamma \parallel b \parallel k) \leftarrow \text{punc.Eval}^{-1}(k_{\text{master}}\{M^*, K_{\text{enc}}^*\}, K_{\text{enc}})$.
 3. If $b = 0$, then output $\text{base.}\Pi^{-1}(k, y)$.
 4. Else ($b = 1$), output $\text{base.}\Pi^{-1}(k, \Gamma^{-1}(y))$.

By the functionality-preserving property of the puncturable PRP, and since the circuits never evaluate the PRP on the punctured point, the circuits from the previous hybrid (which evaluate using the unpunctured key) and those in this hybrid (which evaluate using the punctured key) are functionally

equivalent. Therefore, by iO security,

$$\left\{ \begin{array}{l} (crs, K_{enc}^*) \\ \left\{ \begin{array}{l} k_{master} \leftarrow \{0, 1\}^{\lambda'} \\ k^* \leftarrow \{0, 1\}^{\lambda} \\ M^* := (\Gamma^* \parallel b^* \parallel k^*) \\ K_{enc}^* \leftarrow \text{punc.}\Pi(k_{master}, M^*) \\ \widehat{C_{eval}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{eval}[k_{master}, M^*, K_{enc}^*]) \\ \widehat{C_{inv}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{inv}[k_{master}, M^*, K_{enc}^*]) \\ crs := (\widehat{C_{eval}}, \widehat{C_{inv}}) \end{array} \right. \end{array} \right\} \\ \approx_{\epsilon_{iO}(\lambda')} \left\{ \begin{array}{l} (crs, K_{enc}^*) \\ \left\{ \begin{array}{l} k_{master} \leftarrow \{0, 1\}^{\lambda'} \\ k^* \leftarrow \{0, 1\}^{\lambda} \\ M^* := (\Gamma^* \parallel b^* \parallel k^*) \\ K_{enc}^* \leftarrow \text{punc.}\Pi(k_{master}, M^*) \\ k_{master}\{M^*, K_{enc}^*\} \leftarrow \text{punc.Punc}(k_{master}, M^*, K_{enc}^*) \\ \widehat{C_{eval}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{eval}[k_{master}\{M^*, K_{enc}^*\}, M^*, K_{enc}^*]) \\ \widehat{C_{inv}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{inv}[k_{master}\{M^*, K_{enc}^*\}, M^*, K_{enc}^*]) \\ crs := (\widehat{C_{eval}}, \widehat{C_{inv}}) \end{array} \right. \end{array} \right\} .$$

Hybrid 3: In this hybrid, we replace the encrypted key $K_{enc}^* \leftarrow \text{punc.}\Pi(k_{master}, M^*)$ with a uniformly random domain element $K_{enc}^{\text{rand}} \leftarrow [N']$. By the security of the puncturable PRP, for any fixed $M^* := (\Gamma^* \parallel b^* \parallel k^*)$, the tuple $\{k\{M^*, K_{enc}^*\}, M^*, K_{enc}^*\}$ with an honestly computed evaluation $K_{enc}^* \leftarrow \text{punc.}\Pi(k_{master}, M^*)$ is $\epsilon'_{\text{PRP}}(\lambda', N')$ indistinguishable from one where K_{enc}^* is sampled uniformly at random from $[N']$. This indistinguishability still holds when k^* is omitted from the view. Therefore,

$$\left\{ \begin{array}{l} (crs, K_{enc}^*) \\ \left\{ \begin{array}{l} k_{master} \leftarrow \{0, 1\}^{\lambda'} \\ k^* \leftarrow \{0, 1\}^{\lambda} \\ M^* := (\Gamma^* \parallel b^* \parallel k^*) \\ K_{enc}^* \leftarrow \text{punc.}\Pi(k_{master}, M^*) \\ k_{master}\{M^*, K_{enc}^*\} \leftarrow \text{punc.Punc}(k_{master}, M^*, K_{enc}^*) \\ \widehat{C_{eval}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{eval}[k_{master}\{M^*, K_{enc}^*\}, M^*, K_{enc}^*]) \\ \widehat{C_{inv}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{inv}[k_{master}\{M^*, K_{enc}^*\}, M^*, K_{enc}^*]) \\ crs := (\widehat{C_{eval}}, \widehat{C_{inv}}) \end{array} \right. \end{array} \right\} \\ \approx_{\epsilon'_{\text{PRP}}(\lambda', N')} \left\{ \begin{array}{l} (crs, K_{enc}^*) \\ \left\{ \begin{array}{l} k_{master} \leftarrow \{0, 1\}^{\lambda'} \\ k^* \leftarrow \{0, 1\}^{\lambda} \\ M^* := (\Gamma^* \parallel b^* \parallel k^*) \\ K_{enc}^* \leftarrow [N'] \\ k_{master}\{M^*, K_{enc}^*\} \leftarrow \text{punc.Punc}(k_{master}, M^*) \\ \widehat{C_{eval}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{eval}[k_{master}\{M^*, K_{enc}^*\}, M^*, K_{enc}^*]) \\ \widehat{C_{inv}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{inv}[k_{master}\{M^*, K_{enc}^*\}, M^*, K_{enc}^*]) \\ crs := (\widehat{C_{eval}}, \widehat{C_{inv}}) \end{array} \right. \end{array} \right\} .$$

Hybrid 4: In this hybrid, we replace the punctured master key with the unpunctured original master key. Since the functionality of the circuit remains identical, the two hybrids are indistin-

guishable by iO security.

$$\left\{ \begin{array}{l} \text{(crs, } K_{\text{enc}}^*) \\ \left\{ \begin{array}{l} k_{\text{master}} \leftarrow \{0, 1\}^{\lambda'} \\ k^* \leftarrow \{0, 1\}^{\lambda} \\ M^* := (\Gamma^* \parallel b^* \parallel k^*) \\ K_{\text{enc}}^* \leftarrow [N'] \\ \textcolor{red}{k_{\text{master}}\{M^*, K_{\text{enc}}^*\}} \leftarrow \text{punc.Punc}(k_{\text{master}}, M^*, K_{\text{enc}}^*) \\ \widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{eval}}[\textcolor{red}{k_{\text{master}}\{M^*, K_{\text{enc}}^*\}}, M^*, K_{\text{enc}}^*]) \\ \widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{inv}}[\textcolor{red}{k_{\text{master}}\{M^*, K_{\text{enc}}^*\}}, M^*, K_{\text{enc}}^*]) \\ \text{crs} := (\widehat{C_{\text{eval}}}, \widehat{C_{\text{inv}}}) \end{array} \right\} \end{array} \right\} \\ \approx_{\epsilon_{\text{iO}}(\lambda')} \left\{ \begin{array}{l} \text{(crs, } K_{\text{enc}}^*) \\ \left\{ \begin{array}{l} k_{\text{master}} \leftarrow \{0, 1\}^{\lambda'} \\ k^* \leftarrow \{0, 1\}^{\lambda} \\ M^* := (\Gamma^* \parallel b^* \parallel k^*) \\ K_{\text{enc}}^* \leftarrow [N'] \\ \widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{eval}}[\textcolor{red}{k_{\text{master}}}, M^*, K_{\text{enc}}^*]) \\ \widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{inv}}[\textcolor{red}{k_{\text{master}}}, M^*, K_{\text{enc}}^*]) \\ \text{crs} := (\widehat{C_{\text{eval}}}, \widehat{C_{\text{inv}}}) \end{array} \right\} \end{array} \right\}.$$

Hybrid 5: In this hybrid, instead of hardcoding $M^* = (\Gamma^* \parallel b^* \parallel k^*)$ to compute $\Gamma^*(\text{base}.\Pi(k^*, x^*))$, we hardcode the permuted key $k^{*(\Gamma^*, b^*)} \leftarrow \text{base.Permute}(k^*, \Gamma^*, b^*)$ and compute the output directly as $\text{base}.\Pi(k^{*(\Gamma^*, b^*)}, x)$.

- $C_{\text{eval}}[k_{\text{master}}, \textcolor{red}{k^{*(\Gamma^*, b^*)}}, K_{\text{enc}}^*](K_{\text{enc}}, x)$:
 1. If $K_{\text{enc}} = K_{\text{enc}}^*$, then output $\textcolor{red}{\text{base}.\Pi(k^{*(\Gamma^*, b^*)}, x)}$.
 2. Else, compute $(\Gamma \parallel b \parallel k) \leftarrow \text{punc}.\Pi^{-1}(k_{\text{master}}, K_{\text{enc}})$.
 3. If $b = 0$, then output $\text{base}.\Pi(k, x)$.
 4. Else ($b = 1$), output $\Gamma(\text{base}.\Pi(k, x))$.
- $C_{\text{inv}}[k_{\text{master}}, \textcolor{red}{k^{*(\Gamma^*, b^*)}}, K_{\text{enc}}^*](K_{\text{enc}}, y)$:
 1. If $K_{\text{enc}} = K_{\text{enc}}^*$, then output $\textcolor{red}{\text{base}.\Pi^{-1}(k^{*(\Gamma^*, b^*)}, x)}$.
 2. Else, compute $(\Gamma \parallel b \parallel k) \leftarrow \text{punc}.\Pi^{-1}(k_{\text{master}}, K_{\text{enc}})$.
 3. If $b = 0$, then output $\text{base}.\Pi^{-1}(k, y)$.
 4. Else ($b = 1$), output $\text{base}.\Pi^{-1}(k, \Gamma^{-1}(y))$.

By the correctness of key permutation, the functionality of the circuit remains the same. Therefore,

by the iO security,

$$\left\{ \begin{array}{l} \text{crs}, K_{\text{enc}}^* \\ \left| \begin{array}{l} k_{\text{master}} \leftarrow \{0, 1\}^{\lambda'} \\ k^* \leftarrow \{0, 1\}^{\lambda} \\ \textcolor{red}{M^*} := (\Gamma^* \parallel b^* \parallel k^*) \\ K_{\text{enc}}^* \leftarrow [N'] \\ \widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{eval}}[k_{\text{master}}, \textcolor{red}{M^*}, K_{\text{enc}}^*]) \\ \widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{inv}}[k_{\text{master}}, \textcolor{red}{M^*}, K_{\text{enc}}^*]) \\ \text{crs} := (\widehat{C_{\text{eval}}}, \widehat{C_{\text{inv}}}) \end{array} \right. \end{array} \right\} \\ \approx_{\epsilon_{\text{iO}}(\lambda')} \left\{ \begin{array}{l} \text{crs}, K_{\text{enc}}^* \\ \left| \begin{array}{l} k_{\text{master}} \leftarrow \{0, 1\}^{\lambda'} \\ k^* \leftarrow \{0, 1\}^{\lambda} \\ K_{\text{enc}}^* \leftarrow [N'] \\ \textcolor{red}{k^*(\Gamma^*, b^*)} \leftarrow \text{base.Permute}(k^*, \Gamma^*, b^*) \\ \widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{eval}}[k_{\text{master}}, \textcolor{red}{k^*(\Gamma^*, b^*)}, K_{\text{enc}}^*]) \\ \widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{inv}}[k_{\text{master}}, \textcolor{red}{k^*(\Gamma^*, b^*)}, K_{\text{enc}}^*]) \\ \text{crs} := (\widehat{C_{\text{eval}}}, \widehat{C_{\text{inv}}}) \end{array} \right. \end{array} \right\}.$$

Hybrid 6: In this hybrid, we want to compute the permuted key with a uniformly random swap Γ^{rand} and bit b^{rand} rather than use the fixed Γ^*, b^* .

Claim 9.5.

$$\left\{ \begin{array}{l} \text{crs}, K_{\text{enc}}^* \\ \left| \begin{array}{l} k_{\text{master}} \leftarrow \{0, 1\}^{\lambda'} \\ k^* \leftarrow \{0, 1\}^{\lambda} \\ K_{\text{enc}}^* \leftarrow [N'] \\ k^*(\Gamma^*, b^*) \leftarrow \text{base.Permute}(k^*, \Gamma^*, b^*) \\ \widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{eval}}[k_{\text{master}}, k^*(\Gamma^*, b^*), K_{\text{enc}}^*]) \\ \widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{inv}}[k_{\text{master}}, k^*(\Gamma^*, b^*), K_{\text{enc}}^*]) \\ \text{crs} := (\widehat{C_{\text{eval}}}, \widehat{C_{\text{inv}}}) \end{array} \right. \end{array} \right\} \\ \approx_{O(\epsilon_{\text{PRP}}(\lambda', N) + \epsilon_{\text{iO}}(\lambda'))} \left\{ \begin{array}{l} \text{crs}, K_{\text{enc}}^* \\ \left| \begin{array}{l} k_{\text{master}} \leftarrow \{0, 1\}^{\lambda'} \\ k^* \leftarrow \{0, 1\}^{\lambda} \\ K_{\text{enc}}^* \leftarrow [N'] \\ \Gamma^{\text{rand}} \leftarrow G \\ b^{\text{rand}} \leftarrow \{0, 1\} \\ \textcolor{red}{k^*(\Gamma^{\text{rand}}, b^{\text{rand}})} \leftarrow \text{base.Permute}(k^*, \Gamma^{\text{rand}}, b^{\text{rand}}) \\ \widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{eval}}[k_{\text{master}}, \textcolor{red}{k^*(\Gamma^{\text{rand}}, b^{\text{rand}})}, K_{\text{enc}}^*]) \\ \widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{inv}}[k_{\text{master}}, \textcolor{red}{k^*(\Gamma^{\text{rand}}, b^{\text{rand}})}, K_{\text{enc}}^*]) \\ \text{crs} := (\widehat{C_{\text{eval}}}, \widehat{C_{\text{inv}}}) \end{array} \right. \end{array} \right\}.$$

Proof. Since k^* is sampled uniform at random, and the circuits does not contain any information

about the key other than the permuted key, we get that by the permuted security,

$$\approx_{\epsilon_{\text{PRP}}(\lambda', N)} \left\{ \begin{array}{l} (\text{crs}, K_{\text{enc}}^*) \left| \begin{array}{l} k_{\text{master}} \leftarrow \{0, 1\}^{\lambda'} \\ k^* \leftarrow \{0, 1\}^{\lambda} \\ K_{\text{enc}}^* \leftarrow [N'] \\ k^{*(\Gamma^*, b^*)} \leftarrow \text{base.Permute}(k^*, \Gamma^*, b^*) \\ \widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{eval}}[k_{\text{master}}, k^{*(\Gamma^*, b^*)}, K_{\text{enc}}^*]) \\ \widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{inv}}[k_{\text{master}}, k^{*(\Gamma^*, b^*)}, K_{\text{enc}}^*]) \\ \text{crs} := (\widehat{C_{\text{eval}}}, \widehat{C_{\text{inv}}}) \end{array} \right. \end{array} \right\}.$$

Note that for $b = 0$, the function $\text{base.}\Pi(k^{*(\Gamma^*, 0)}, \cdot)$ is functionally equivalent to $\text{base.}\Pi(k^*, \cdot)$. Similarly, for any $\Gamma^{\text{rand}} \in G$, the function $\text{base.}\Pi(k^{*(\Gamma^{\text{rand}}, 0)}, \cdot)$ is equivalent to $\text{base.}\Pi(k^*, \cdot)$. Therefore, $\text{base.}\Pi(k^{*(\Gamma^*, 0)}, \cdot)$ and $\text{base.}\Pi(k^{*(\Gamma^{\text{rand}}, 0)}, \cdot)$ are functionally equivalent. We get that by iO security,

$$\approx_{\epsilon_{\text{iO}}(\lambda')} \left\{ \begin{array}{l} (\text{crs}, K_{\text{enc}}^*) \left| \begin{array}{l} k_{\text{master}} \leftarrow \{0, 1\}^{\lambda'} \\ k^* \leftarrow \{0, 1\}^{\lambda} \\ K_{\text{enc}}^* \leftarrow [N'] \\ \textcolor{red}{k^{*(\Gamma^*, 0)}} \leftarrow \text{base.Permute}(k^*, \Gamma^*, 0) \\ \widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{eval}}[k_{\text{master}}, k^{*(\Gamma^*, 0)}, K_{\text{enc}}^*]) \\ \widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{inv}}[k_{\text{master}}, k^{*(\Gamma^*, 0)}, K_{\text{enc}}^*]) \\ \text{crs} := (\widehat{C_{\text{eval}}}, \widehat{C_{\text{inv}}}) \end{array} \right. \end{array} \right\}.$$

$$\approx_{\epsilon_{\text{iO}}(\lambda')} \left\{ \begin{array}{l} (\text{crs}, K_{\text{enc}}^*) \left| \begin{array}{l} k_{\text{master}} \leftarrow \{0, 1\}^{\lambda'} \\ k^* \leftarrow \{0, 1\}^{\lambda} \\ \textcolor{red}{\Gamma^{\text{rand}}} \leftarrow G \\ K_{\text{enc}}^* \leftarrow [N'] \\ \textcolor{red}{k^{*(\Gamma^{\text{rand}}, 0)}} \leftarrow \text{base.Permute}(k^*, \Gamma^{\text{rand}}, 0) \\ \widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{eval}}[k_{\text{master}}, k^{*(\Gamma^{\text{rand}}, 0)}, K_{\text{enc}}^*]) \\ \widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{inv}}[k_{\text{master}}, k^{*(\Gamma^{\text{rand}}, 0)}, K_{\text{enc}}^*]) \\ \text{crs} := (\widehat{C_{\text{eval}}}, \widehat{C_{\text{inv}}}) \end{array} \right. \end{array} \right\}.$$

By the permuted key security,

$$\begin{aligned}
& \left\{ (crs, K_{enc}^*) \left| \begin{array}{l} k_{master} \leftarrow \{0,1\}^{\lambda'} \\ k^* \leftarrow \{0,1\}^\lambda \\ \Gamma^{rand} \leftarrow G \\ K_{enc}^* \leftarrow [N'] \\ k^{*(\Gamma^{rand},0)} \leftarrow \text{base.Permute}(k^*, \Gamma^{rand}, 0) \\ \widehat{C_{eval}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{eval}[k_{master}, k^{*(\Gamma^{rand},0)}, K_{enc}^*]) \\ \widehat{C_{inv}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{inv}[k_{master}, k^{*(\Gamma^{rand},0)}, K_{enc}^*]) \\ crs := (\widehat{C_{eval}}, \widehat{C_{inv}}) \end{array} \right. \right\} \\
& \approx_{\epsilon_{PRP}(\lambda', N)} \left\{ (crs, K_{enc}^*) \left| \begin{array}{l} k_{master} \leftarrow \{0,1\}^{\lambda'} \\ k^* \leftarrow \{0,1\}^\lambda \\ \Gamma^{rand} \leftarrow G \\ \textcolor{red}{b^{rand}} \leftarrow \{0,1\} \\ K_{enc}^* \leftarrow [N'] \\ k^{*(\Gamma^{rand}, b^{rand})} \leftarrow \text{base.Permute}(k^*, \Gamma^{rand}, \textcolor{red}{b^{rand}}) \\ \widehat{C_{eval}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{eval}[k_{master}, \textcolor{red}{k^{*(\Gamma^{rand}, b^{rand})}}, K_{enc}^*]) \\ \widehat{C_{inv}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{inv}[k_{master}, \textcolor{red}{k^{*(\Gamma^{rand}, b^{rand})}}, K_{enc}^*]) \\ crs := (\widehat{C_{eval}}, \widehat{C_{inv}}) \end{array} \right. \right\}.
\end{aligned}$$

Overall, we get that,

$$\begin{aligned}
& \left\{ (crs, K_{enc}^*) \left| \begin{array}{l} k_{master} \leftarrow \{0,1\}^{\lambda'} \\ k^* \leftarrow \{0,1\}^\lambda \\ K_{enc}^* \leftarrow [N'] \\ \textcolor{red}{k^{*(\Gamma^*, b^*)}} \leftarrow \text{base.Permute}(k^*, \Gamma^*, b^*) \\ \widehat{C_{eval}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{eval}[k_{master}, k^{*(\Gamma^*, b^*)}, K_{enc}^*]) \\ \widehat{C_{inv}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{inv}[k_{master}, k^{*(\Gamma^*, b^*)}, K_{enc}^*]) \\ crs := (\widehat{C_{eval}}, \widehat{C_{inv}}) \end{array} \right. \right\} \\
& \approx_{O(\epsilon_{PRP}(\lambda', N) + \text{iO}(\lambda'))} \left\{ (crs, K_{enc}^*) \left| \begin{array}{l} k_{master} \leftarrow \{0,1\}^{\lambda'} \\ k^* \leftarrow \{0,1\}^\lambda \\ K_{enc}^* \leftarrow [N'] \\ \Gamma^{rand} \leftarrow G \\ \textcolor{red}{b^{rand}} \leftarrow \{0,1\} \\ \textcolor{red}{k^{*(\Gamma^{rand}, b^{rand})}} \leftarrow \text{base.Permute}(k^*, \Gamma^{rand}, \textcolor{red}{b^{rand}}) \\ \widehat{C_{eval}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{eval}[k_{master}, \textcolor{red}{k^{*(\Gamma^{rand}, b^{rand})}}, K_{enc}^*]) \\ \widehat{C_{inv}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{inv}[k_{master}, \textcolor{red}{k^{*(\Gamma^{rand}, b^{rand})}}, K_{enc}^*]) \\ crs := (\widehat{C_{eval}}, \widehat{C_{inv}}) \end{array} \right. \right\},
\end{aligned}$$

as required. $\square$

Hybrid 7: In this hybrid, we replace the hardcoded permuted key $k^{*(\Gamma^{rand}, b^{rand})} \leftarrow \text{base.Permute}(k^*, \Gamma^{rand}, b^{rand})$ by $M^* = (\Gamma^{rand} \parallel b^{rand} \parallel k^*)$. The special branches evaluate the base permutation, composed with Γ^{rand} when $b^{rand} = 1$, and its inverse. The circuits are then defined as follows.

- $C_{\text{eval}}[\mathbf{k}_{\text{master}}, M^*, K_{\text{enc}}^*](K_{\text{enc}}, x)$:
 1. If $K_{\text{enc}} = K_{\text{enc}}^*$, then set $(\Gamma \parallel b \parallel \mathbf{k}) := M^*$.
 2. Else, compute $(\Gamma \parallel b \parallel \mathbf{k}) \leftarrow \text{punc}.\Pi^{-1}(\mathbf{k}_{\text{master}}, K_{\text{enc}})$.
 3. If $b = 0$, then output $\text{base}.\Pi(\mathbf{k}, x)$.
 4. Else ($b = 1$), output $\Gamma(\text{base}.\Pi(\mathbf{k}, x))$.
- $C_{\text{inv}}[\mathbf{k}_{\text{master}}, M^*, K_{\text{enc}}^*](K_{\text{enc}}, y)$:
 1. If $K_{\text{enc}} = K_{\text{enc}}^*$, then set $(\Gamma \parallel b \parallel \mathbf{k}) := M^*$.
 2. Else, compute $(\Gamma \parallel b \parallel \mathbf{k}) \leftarrow \text{punc}.\Pi^{-1}(\mathbf{k}_{\text{master}}, K_{\text{enc}})$.
 3. If $b = 0$, then output $\text{base}.\Pi^{-1}(\mathbf{k}, y)$.
 4. Else ($b = 1$), output $\text{base}.\Pi^{-1}(\mathbf{k}, \Gamma^{-1}(y))$.

By the correctness of key permutation, the functionality of the circuit remains the same. Therefore, by the iO security,

$$\begin{aligned}
 & \left\{ (\text{crs}, K_{\text{enc}}^*) \left| \begin{array}{l} \mathbf{k}_{\text{master}} \leftarrow \{0, 1\}^{\lambda'} \\ \mathbf{k}^* \leftarrow \{0, 1\}^{\lambda} \\ K_{\text{enc}}^* \leftarrow [N'] \\ \Gamma^{\text{rand}} \leftarrow G \\ b^{\text{rand}} \leftarrow \{0, 1\} \\ \mathbf{k}^*(\Gamma^{\text{rand}}, b^{\text{rand}}) \leftarrow \text{base}.\text{Permute}(\mathbf{k}^*, \Gamma^{\text{rand}}, b^{\text{rand}}) \\ \widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{eval}}[\mathbf{k}_{\text{master}}, \mathbf{k}^*(\Gamma^{\text{rand}}, b^{\text{rand}}), K_{\text{enc}}^*]) \\ \widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{inv}}[\mathbf{k}_{\text{master}}, \mathbf{k}^*(\Gamma^{\text{rand}}, b^{\text{rand}}), K_{\text{enc}}^*]) \\ \text{crs} := (\widehat{C_{\text{eval}}}, \widehat{C_{\text{inv}}}) \end{array} \right. \right\} \\
 & \approx_{\epsilon_{\text{iO}}(\lambda')} \left\{ (\text{crs}, K_{\text{enc}}^*) \left| \begin{array}{l} \mathbf{k}_{\text{master}} \leftarrow \{0, 1\}^{\lambda'} \\ \mathbf{k}^* \leftarrow \{0, 1\}^{\lambda} \\ K_{\text{enc}}^* \leftarrow [N'] \\ \Gamma^{\text{rand}} \leftarrow G \\ b^{\text{rand}} \leftarrow \{0, 1\} \\ M^* := (\Gamma^{\text{rand}} \parallel b^{\text{rand}} \parallel \mathbf{k}^*) \\ \widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{eval}}[\mathbf{k}_{\text{master}}, M^*, K_{\text{enc}}^*]) \\ \widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{inv}}[\mathbf{k}_{\text{master}}, M^*, K_{\text{enc}}^*]) \\ \text{crs} := (\widehat{C_{\text{eval}}}, \widehat{C_{\text{inv}}}) \end{array} \right. \right\}.
 \end{aligned}$$

Hybrid 8: Let $\mathbf{k}_{\text{master}}\{M^*, K_{\text{enc}}^*\}$ be the master key $\mathbf{k}_{\text{master}}$ punctured at the points (M^*, K_{enc}^*) . Then, we define $C_{\text{eval}}[\mathbf{k}_{\text{master}}\{M^*, K_{\text{enc}}^*\}, M^*, K_{\text{enc}}^*]$ and $C_{\text{inv}}[\mathbf{k}_{\text{master}}\{M^*, K_{\text{enc}}^*\}, M^*, K_{\text{enc}}^*]$ as follows.

- $C_{\text{eval}}[\mathbf{k}_{\text{master}}\{M^*, K_{\text{enc}}^*\}, M^*, K_{\text{enc}}^*](K_{\text{enc}}, x)$:
 1. If $K_{\text{enc}} = K_{\text{enc}}^*$, then set $(\Gamma \parallel b \parallel \mathbf{k}) := M^*$.
 2. Else, compute $(\Gamma \parallel b \parallel \mathbf{k}) \leftarrow \text{punc}.\text{Eval}^{-1}(\mathbf{k}_{\text{master}}\{M^*, K_{\text{enc}}^*\}, K_{\text{enc}})$.
 3. If $b = 0$, then output $\text{base}.\Pi(\mathbf{k}, x)$.
 4. Else ($b = 1$), output $\Gamma(\text{base}.\Pi(\mathbf{k}, x))$.
- $C_{\text{inv}}[\mathbf{k}_{\text{master}}\{M^*, K_{\text{enc}}^*\}, M^*, K_{\text{enc}}^*](K_{\text{enc}}, y)$:
 1. If $K_{\text{enc}} = K_{\text{enc}}^*$, then set $(\Gamma \parallel b \parallel \mathbf{k}) := M^*$.

2. Else, compute $(\Gamma \parallel b \parallel k) \leftarrow \text{punc.Eval}^{-1}(k_{\text{master}}\{M^*, K_{\text{enc}}^*\}, K_{\text{enc}})$.
3. If $b = 0$, then output $\text{base}.\Pi^{-1}(k, y)$.
4. Else ($b = 1$), output $\text{base}.\Pi^{-1}(k, \Gamma^{-1}(y))$.

By the functionality-preserving property of the puncturable PRP, and since the circuits never evaluate the PRP on the punctured point, the circuits from the previous hybrid and those in this hybrid are functionally equivalent. Therefore, by iO security,

$$\begin{aligned}
& \left\{ (crs, K_{\text{enc}}^*) \left| \begin{array}{l} k_{\text{master}} \leftarrow \{0, 1\}^{\lambda'} \\ k^* \leftarrow \{0, 1\}^{\lambda} \\ K_{\text{enc}}^* \leftarrow [N'] \\ \Gamma^{\text{rand}} \leftarrow G \\ b^{\text{rand}} \leftarrow \{0, 1\} \\ M^* := (\Gamma^{\text{rand}} \parallel b^{\text{rand}} \parallel k^*) \\ \widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{eval}}[k_{\text{master}}, M^*, K_{\text{enc}}^*]) \\ \widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{inv}}[k_{\text{master}}, M^*, K_{\text{enc}}^*]) \\ crs := (\widehat{C_{\text{eval}}}, \widehat{C_{\text{inv}}}) \end{array} \right. \right\} \\
& \approx_{\epsilon_{\text{iO}}(\lambda')} \left\{ (crs, K_{\text{enc}}^*) \left| \begin{array}{l} k_{\text{master}} \leftarrow \{0, 1\}^{\lambda'} \\ k^* \leftarrow \{0, 1\}^{\lambda} \\ K_{\text{enc}}^* \leftarrow [N'] \\ \Gamma^{\text{rand}} \leftarrow G \\ b^{\text{rand}} \leftarrow \{0, 1\} \\ M^* := (\Gamma^{\text{rand}} \parallel b^{\text{rand}} \parallel k^*) \\ k_{\text{master}}\{M^*, K_{\text{enc}}^*\} \leftarrow \text{punc.Punc}(k_{\text{master}}, M^*, K_{\text{enc}}^*) \\ \widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{eval}}[k_{\text{master}}\{M^*, K_{\text{enc}}^*\}, M^*, K_{\text{enc}}^*]) \\ \widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{inv}}[k_{\text{master}}\{M^*, K_{\text{enc}}^*\}, M^*, K_{\text{enc}}^*]) \\ crs := (\widehat{C_{\text{eval}}}, \widehat{C_{\text{inv}}}) \end{array} \right. \right\}.
\end{aligned}$$

Hybrid 9: Note that in the experiment of the previous hybrid, M^* is just sampled uniformly at random. In this hybrid, we replace the uniformly random value M^* with the preimage of K_{enc}^* : $M^* \leftarrow \text{punc}.\Pi^{-1}(k_{\text{master}}, K_{\text{enc}}^*)$. By the security of the puncturable PRP, for any $K_{\text{enc}}^* \in [N']$, the tuple $\{k\{M^*, K_{\text{enc}}^*\}, M^*, K_{\text{enc}}^*\}$ with uniformly random value of M^* is $\epsilon'_{\text{PRP}}(\lambda', N')$ indistinguishable

from one where $K_{\text{enc}}^* \leftarrow \text{punc}.\Pi(k_{\text{master}}, M^*)$. Therefore,

$$\left\{ \begin{array}{l} \text{crs}, K_{\text{enc}}^* \\ \left| \begin{array}{l} k_{\text{master}} \leftarrow \{0, 1\}^{\lambda'} \\ \mathbf{k}^* \leftarrow \{0, 1\}^{\lambda} \\ K_{\text{enc}}^* \leftarrow [N'] \\ \Gamma^{\text{rand}} \leftarrow G \\ b^{\text{rand}} \leftarrow \{0, 1\} \\ M^* := (\Gamma^{\text{rand}} \parallel b^{\text{rand}} \parallel \mathbf{k}^*) \\ k_{\text{master}}\{M^*, K_{\text{enc}}^*\} \leftarrow \text{punc}.\text{Punc}(k_{\text{master}}, M^*, K_{\text{enc}}^*) \\ \widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{eval}}[k_{\text{master}}\{M^*, K_{\text{enc}}^*\}, M^*, K_{\text{enc}}^*]) \\ \widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{inv}}[k_{\text{master}}\{M^*, K_{\text{enc}}^*\}, M^*, K_{\text{enc}}^*]) \\ \text{crs} := (\widehat{C_{\text{eval}}}, \widehat{C_{\text{inv}}}) \end{array} \right. \end{array} \right\} \\ \approx_{\epsilon'_{\text{PRP}}(\lambda', N')} \left\{ \begin{array}{l} \text{crs}, K_{\text{enc}}^* \\ \left| \begin{array}{l} k_{\text{master}} \leftarrow \{0, 1\}^{\lambda'} \\ K_{\text{enc}}^* \leftarrow [N'] \\ M^* \leftarrow \text{punc}.\Pi^{-1}(k_{\text{master}}, K_{\text{enc}}^*) \\ k_{\text{master}}\{M^*, K_{\text{enc}}^*\} \leftarrow \text{punc}.\text{Punc}(k_{\text{master}}, M^*, K_{\text{enc}}^*) \\ \widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{eval}}[k_{\text{master}}\{M^*, K_{\text{enc}}^*\}, M^*, K_{\text{enc}}^*]) \\ \widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{inv}}[k_{\text{master}}\{M^*, K_{\text{enc}}^*\}, M^*, K_{\text{enc}}^*]) \\ \text{crs} := (\widehat{C_{\text{eval}}}, \widehat{C_{\text{inv}}}) \end{array} \right. \end{array} \right\} .$$

Hybrid 10: Here, we return to generating the crs honestly via the Gen algorithm, while continuing to sample $K_{\text{enc}}^* \leftarrow [N']$ uniformly at random. Since M^* is computed as the preimage of K_{enc}^* (i.e., $M^* \leftarrow \text{punc}.\Pi^{-1}(k_{\text{master}}, K_{\text{enc}}^*)$), the circuits $C_{\text{eval}}[k_{\text{master}}\{M^*, K_{\text{enc}}^*\}, M^*, K_{\text{enc}}^*]$ and $C_{\text{inv}}[k_{\text{master}}\{M^*, K_{\text{enc}}^*\}, M^*, K_{\text{enc}}^*]$ are functionally equivalent to $C_{\text{eval}}[k_{\text{master}}]$ and $C_{\text{inv}}[k_{\text{master}}]$ as defined in Construction 9.4. Therefore, by iO security.

$$\left\{ \begin{array}{l} \text{crs}, K_{\text{enc}}^* \\ \left| \begin{array}{l} k_{\text{master}} \leftarrow \{0, 1\}^{\lambda'} \\ K_{\text{enc}}^* \leftarrow [N'] \\ M^* \leftarrow \text{punc}.\Pi^{-1}(k_{\text{master}}, K_{\text{enc}}^*) \\ k_{\text{master}}\{M^*, K_{\text{enc}}^*\} \leftarrow \text{punc}.\text{Punc}(k_{\text{master}}, M^*, K_{\text{enc}}^*) \\ \widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{eval}}[k_{\text{master}}\{M^*, K_{\text{enc}}^*\}, M^*, K_{\text{enc}}^*]) \\ \widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{inv}}[k_{\text{master}}\{M^*, K_{\text{enc}}^*\}, M^*, K_{\text{enc}}^*]) \\ \text{crs} := (\widehat{C_{\text{eval}}}, \widehat{C_{\text{inv}}}) \end{array} \right. \end{array} \right\} \\ \approx_{\epsilon_{\text{iO}}(\lambda')} \left\{ \begin{array}{l} \text{crs}, K_{\text{enc}}^* \\ \left| \begin{array}{l} k_{\text{master}} \leftarrow \{0, 1\}^{\lambda'} \\ K_{\text{enc}}^* \leftarrow [N'] \\ \widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{eval}}[k_{\text{master}}]) \\ \widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, C_{\text{inv}}[k_{\text{master}}]) \\ \text{crs} := (\widehat{C_{\text{eval}}}, \widehat{C_{\text{inv}}}) \end{array} \right. \end{array} \right\} .$$

By construction, we get that,

$$\left\{ \begin{array}{l|l} (\text{crs}, K_{\text{enc}}^*) & \begin{array}{l} k_{\text{master}} \leftarrow \{0, 1\}^{\lambda'} \\ K_{\text{enc}}^* \leftarrow [N'] \\ \widehat{C_{\text{eval}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, \textcolor{red}{C_{\text{eval}}[k_{\text{master}}]}) \\ \widehat{C_{\text{inv}}} \leftarrow \text{iO}(1^{\lambda'}, 1^s, \textcolor{red}{C_{\text{inv}}[k_{\text{master}}]}) \\ \text{crs} := (\widehat{C_{\text{eval}}}, \widehat{C_{\text{inv}}}) \end{array} \end{array} \right\} \\ \equiv \left\{ \begin{array}{l|l} (\text{crs}, K_{\text{enc}}^*) & \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ K_{\text{enc}}^* \leftarrow [N'] \end{array} \end{array} \right\} .$$

Overall, we get that for every $\Gamma^* \in G$ and $b^* \in \{0, 1\}$,

$$\left\{ \begin{array}{l|l} (\text{crs}, K_{\text{enc}}^*) & \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ k^* \leftarrow \{0, 1\}^\lambda \\ K_{\text{enc}}^* \leftarrow \text{Permute}(\text{pv}, (\Gamma^* \parallel b^* \parallel k^*)) \end{array} \end{array} \right\} \\ \approx_{\epsilon'} \left\{ \begin{array}{l|l} (\text{crs}, K_{\text{enc}}^*) & \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ K_{\text{enc}}^* \leftarrow [N'] \end{array} \end{array} \right\} ,$$

where $\epsilon' := O(\epsilon_{\text{PRP}}(\lambda', N) + \epsilon'_{\text{PRP}}(\lambda', N') + \epsilon_{\text{iO}}(\lambda'))$. Therefore, for every $\Gamma^* \in G$ and $b^* \in \{0, 1\}$, and for every PPT adversary $\mathbf{A}$,

$$\Pr \left[\begin{array}{l|l} c = c' & \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ c \leftarrow \{0, 1\} \\ k^* \leftarrow \{0, 1\}^{\lambda'} \\ K_{\text{enc}}^* \leftarrow \begin{cases} \text{Permute}(\text{pv}, k^*, \Gamma^*, b^*) & \text{if } c = 0 \\ [N'] & \text{if } c = 1 \end{cases} \\ c' \leftarrow \mathbf{A}(\text{crs}, K_{\text{enc}}^*) \end{array} \end{array} \right] \leq \frac{1}{2} + \epsilon' . \quad (24)$$

This completes the proof of non-adaptive security. We now turn to proving adaptive security.

From non-adaptive to adaptive security. Let $\mathbf{A}' := (\mathbf{A}'_1, \mathbf{A}'_2)$ be a PPT adversary to the new scheme, and let $\nu(\lambda, N)$ be its advantage. Formally,

$$\Pr \left[\begin{array}{l|l} c = c' & \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ (\Gamma^*, b^*, \text{aux}) \leftarrow \mathbf{A}'_1(\text{crs}) \\ c \leftarrow \{0, 1\} \\ k^* \leftarrow \{0, 1\}^{\lambda'} \\ K_{\text{enc}}^* \leftarrow \begin{cases} \text{Permute}(\text{pv}, k^*, \Gamma^*, b^*) & \text{if } c = 0 \\ [N'] & \text{if } c = 1 \end{cases} \\ c' \leftarrow \mathbf{A}'_2(\text{crs}, K_{\text{enc}}^*, \text{aux}) \end{array} \end{array} \right] = \frac{1}{2} + \nu(\lambda, N) .$$

By the law of total probability,

$$\begin{aligned}
& \Pr \left[c = c' \right] \\
&= \sum_{\Gamma' \in G, b' \in \{0,1\}} \Pr \left[\begin{array}{c} \wedge \Gamma^* = \Gamma' \\ \wedge b^* = b' \end{array} \right] \\
&\quad \cdot \Pr \left[\begin{array}{c} c = c' \\ \text{conditioned on} \\ \Gamma^* = \Gamma' \\ \wedge b^* = b' \end{array} \right] \\
& \left[\begin{array}{c} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ (\Gamma^*, b^*, \text{aux}) \leftarrow \mathbf{A}'_1(\text{crs}) \\ c \leftarrow \{0, 1\} \\ k^* \leftarrow \{0, 1\}^{\lambda'} \\ K_{\text{enc}}^* \leftarrow \begin{cases} \text{Permute}(\text{pv}, k^*, \Gamma^*, b^*) & \text{if } c = 0 \\ [N'] & \text{if } c = 1 \end{cases} \\ c' \leftarrow \mathbf{A}'_2(\text{crs}, K_{\text{enc}}^*, \text{aux}) \end{array} \right] \\
& \left[\begin{array}{c} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ (\Gamma^*, b^*, \text{aux}) \leftarrow \mathbf{A}'_1(\text{crs}) \\ c \leftarrow \{0, 1\} \\ k^* \leftarrow \{0, 1\}^{\lambda'} \\ K_{\text{enc}}^* \leftarrow \begin{cases} \text{Permute}(\text{pv}, k^*, \Gamma^*, b^*) & \text{if } c = 0 \\ [N'] & \text{if } c = 1 \end{cases} \\ c' \leftarrow \mathbf{A}'_2(\text{crs}, K_{\text{enc}}^*, \text{aux}) \end{array} \right] \\
& \left[\begin{array}{c} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ (\Gamma^*, b^*, \text{aux}) \leftarrow \mathbf{A}'_1(\text{crs}) \\ c \leftarrow \{0, 1\} \\ k^* \leftarrow \{0, 1\}^{\lambda'} \\ K_{\text{enc}}^* \leftarrow \begin{cases} \text{Permute}(\text{pv}, k^*, \Gamma^*, b^*) & \text{if } c = 0 \\ [N'] & \text{if } c = 1 \end{cases} \\ c' \leftarrow \mathbf{A}'_2(\text{crs}, K_{\text{enc}}^*, \text{aux}) \end{array} \right] .
\end{aligned}$$

For every $\Gamma' \in G$ and $b' \in \{0, 1\}$, let $P[\Gamma', b']$ be the probability that the adversary outputs $\Gamma^* = \Gamma' \wedge b^* = b'$ in the above experiment, and let $\nu[\Gamma', b']$ be the advantage of the adversary conditioned on $\Gamma^* = \Gamma' \wedge b^* = b'$ such that the success probability is $\frac{1}{2} + \nu[\Gamma', b']$. Formally,

$$\begin{aligned}
P[\Gamma', b'] &:= \Pr \left[\begin{array}{c} \wedge \Gamma^* = \Gamma' \\ \wedge b^* = b' \end{array} \right] \\
\nu[\Gamma', b'] &:= \Pr \left[\begin{array}{c} c = c' \\ \text{conditioned on} \\ \Gamma^* = \Gamma' \\ \wedge b^* = b' \end{array} \right] \\
& \left[\begin{array}{c} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ (\Gamma^*, b^*, \text{aux}) \leftarrow \mathbf{A}'_1(\text{crs}) \\ c \leftarrow \{0, 1\} \\ k^* \leftarrow \{0, 1\}^{\lambda'} \\ K_{\text{enc}}^* \leftarrow \begin{cases} \text{Permute}(\text{pv}, k^*, \Gamma^*, b^*) & \text{if } c = 0 \\ [N'] & \text{if } c = 1 \end{cases} \\ c' \leftarrow \mathbf{A}'_2(\text{crs}, K_{\text{enc}}^*, \text{aux}) \end{array} \right] \\
& \left[\begin{array}{c} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ (\Gamma^*, b^*, \text{aux}) \leftarrow \mathbf{A}'_1(\text{crs}) \\ c \leftarrow \{0, 1\} \\ k^* \leftarrow \{0, 1\}^{\lambda'} \\ K_{\text{enc}}^* \leftarrow \begin{cases} \text{Permute}(\text{pv}, k^*, \Gamma^*, b^*) & \text{if } c = 0 \\ [N'] & \text{if } c = 1 \end{cases} \\ c' \leftarrow \mathbf{A}'_2(\text{crs}, K_{\text{enc}}^*, \text{aux}) \end{array} \right] - \frac{1}{2} .
\end{aligned}$$

By the above equation,

$$\begin{aligned}
& \Pr \left[c = c' \mid \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ (\Gamma^*, b^*, \text{aux}) \leftarrow \mathbf{A}'_1(\text{crs}) \\ c \leftarrow \{0, 1\} \\ \mathbf{k}^* \leftarrow \{0, 1\}^{\lambda'} \\ K_{\text{enc}}^* \leftarrow \begin{cases} \text{Permute}(\text{pv}, \mathbf{k}^*, \Gamma^*, b^*) & \text{if } c = 0 \\ [N'] & \text{if } c = 1 \end{cases} \\ c' \leftarrow \mathbf{A}'_2(\text{crs}, K_{\text{enc}}^*, \text{aux}) \end{array} \right] \\
&= \sum_{\Gamma' \in G, b' \in \{0, 1\}} P[\Gamma', b'] \cdot \left(\frac{1}{2} + \nu[\Gamma', b'] \right) \\
&= \frac{1}{2} + \sum_{\Gamma' \in G, b' \in \{0, 1\}} P[\Gamma', b'] \cdot \nu[\Gamma', b'] .
\end{aligned}$$

Note that since the above equals $\frac{1}{2} + \nu(\lambda, N)$, we get that,

$$\sum_{\Gamma' \in G, b' \in \{0, 1\}} P[\Gamma', b'] \cdot \nu[\Gamma', b'] = \nu(\lambda, N) .$$

By an averaging argument, it holds that,

$$\max_{\Gamma' \in G, b' \in \{0, 1\}} P[\Gamma', b'] \cdot \nu[\Gamma', b'] \geq \frac{\nu(\lambda, N)}{2 \cdot |G|} .$$

Fix some $\Gamma' \in G$ and $b' \in \{0, 1\}$ that satisfies the above. It holds that,

$$P[\Gamma', b'] \cdot \nu[\Gamma', b'] \geq \frac{\nu(\lambda, N)}{2 \cdot |G|} . \tag{25}$$

We now use $\mathbf{A}'$ to construct the following non-adaptive PPT adversary $\mathbf{A}$ against the scheme.

1. Given an input $(\text{crs}, K_{\text{enc}}^*)$.
2. Emulate $(\Gamma^*, b^*, \text{aux}) \leftarrow \mathbf{A}'_1(\text{crs})$.
3. If $(\Gamma^* = \Gamma') \wedge (b^* = b')$ then,
 - (a) Emulate $c' \leftarrow \mathbf{A}'_2(\text{crs}, K_{\text{enc}}^*, \text{aux})$.
 - (b) Output c' .
4. Else, output $c' \leftarrow \{0, 1\}$.

Note that the success probability of the adversary, given that $(\Gamma^* \neq \Gamma') \vee (b^* \neq b')$ is $\frac{1}{2}$. By the law

of total probability,

$$\begin{aligned}
& \Pr \left[c = c' \mid \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ c \leftarrow \{0, 1\} \\ \mathbf{k}^* \leftarrow \{0, 1\}^{\lambda'} \\ K_{\text{enc}}^* \leftarrow \begin{cases} \text{Permute}(\text{pv}, \mathbf{k}^*, \Gamma^*, b^*) & \text{if } c = 0 \\ [N'] & \text{if } c = 1 \end{cases} \\ c' \leftarrow \mathbf{A}(\text{crs}, K_{\text{enc}}^*) \end{array} \right] \\
&= (1 - P[\Gamma', b']) \cdot \frac{1}{2} + P[\Gamma', b'] \cdot \left(\frac{1}{2} + \nu[\Gamma', b'] \right) \\
&= \frac{1}{2} + P[\Gamma', b'] \cdot \nu[\Gamma', b'] \\
&\geq \frac{1}{2} + \frac{\nu(\lambda, N)}{2 \cdot |G|} ,
\end{aligned}$$

where the inequality is by Equation 25. On the other hand, by the non-adaptive security (Equation 24), it holds that,

$$\Pr \left[c = c' \mid \begin{array}{l} (\text{crs}, \text{pv}) \leftarrow \text{Gen}(1^\lambda, N) \\ c \leftarrow \{0, 1\} \\ \mathbf{k}^* \leftarrow \{0, 1\}^{\lambda'} \\ K_{\text{enc}}^* \leftarrow \begin{cases} \text{Permute}(\text{pv}, \mathbf{k}^*, \Gamma^*, b^*) & \text{if } c = 0 \\ [N'] & \text{if } c = 1 \end{cases} \\ c' \leftarrow \mathbf{A}(\text{crs}, K_{\text{enc}}^*) \end{array} \right] \leq \frac{1}{2} + \epsilon' ,$$

where $\epsilon' = O(\epsilon(\lambda, N) + \epsilon'_{\text{PRP}}(\lambda', N') + \epsilon_{\text{iO}}(\lambda'))$. Therefore, by the above two equations,

$$\frac{1}{2} + \frac{\nu(\lambda, N)}{2 \cdot |G|} \leq \frac{1}{2} + \epsilon' .$$

By rearranging terms, we obtain,

$$\begin{aligned}
& \nu(\lambda, N) \\
& \leq 2 \cdot |G| \cdot \epsilon' \\
& \leq |G| \cdot O(\epsilon_{\text{PRP}}(\lambda', N) + \epsilon'_{\text{PRP}}(\lambda', N') + \epsilon_{\text{iO}}(\lambda')) \\
& \leq |G| \cdot O(\epsilon_{\text{PRP}}(\lambda', N) + \epsilon_{\text{PRP}}(\lambda', N') + 1/N' + \epsilon_{\text{PRG}}(N') + \epsilon_{\text{iO}}(\lambda')) \\
& \leq |G| \cdot O(\epsilon_{\text{PRP}}(\lambda', N) + \epsilon_{\text{PRP}}(\lambda', N') + 1/2^{\lambda'} + \epsilon_{\text{PRG}}(2^{\lambda'}) + \epsilon_{\text{iO}}(\lambda')) .
\end{aligned}$$

where the second inequality is by the definition of ϵ' , the third inequality is by $\epsilon'_{\text{PRP}}(\lambda', N') \leq O(1/N' + \epsilon_{\text{PRG}}(N') + \epsilon_{\text{iO}}(\lambda') + \epsilon_{\text{PRP}}(\lambda', N'))$, and the fourth inequality holds since $N' < 2^{\lambda'}$. Since $\lambda' = (\lceil \log |G| \rceil \cdot \lambda)^{\frac{1}{\epsilon}}$, and by the ϵ sub-exponential security of the PRP, PRG and iO, we get that,

$$\begin{aligned}
& \nu(\lambda, N) \\
& \leq |G| \cdot O(\epsilon_{\text{PRP}}(\lambda', N) + \epsilon_{\text{PRP}}(\lambda', N') + 1/2^{\lambda'} + \epsilon_{\text{PRG}}(2^{\lambda'}) + \epsilon_{\text{iO}}(\lambda')) \\
& \leq |G| \cdot O(2^{-\lceil \log |G| \rceil \cdot \lambda}) \\
& \leq O(2^{-\lambda}) = \text{negl}(\lambda) ,
\end{aligned}$$

which completes the proof. $\square$

References

- [AS16] Gilad Asharov and Gil Segev. “On Constructing One-Way Permutations from Indistinguishability Obfuscation”. In: *Theory of Cryptography Conference*. Vol. 9563. Lecture Notes in Computer Science. Springer, 2016, pp. 512–541.
- [BBO07] Mihir Bellare, Alexandra Boldyreva, and Adam O’Neill. “Deterministic and Efficiently Searchable Encryption”. In: *Advances in Cryptology, CRYPTO 2007*. Vol. 4622. Lecture Notes in Computer Science. Springer, 2007, pp. 535–552. DOI: 10.1007/978-3-540-74143-5_30.
- [BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. “Non-interactive Zero-knowledge and Its Applications”. In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*. STOC ’88. 1988, pp. 103–112.
- [BGIRSVY01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. “On the (Im)possibility of Obfuscating Programs”. In: *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*. Ed. by Joe Kilian. Vol. 2139. Lecture Notes in Computer Science. Springer, 2001, pp. 1–18.
- [BKPW12] Mihir Bellare, Eike Kiltz, Chris Peikert, and Brent Waters. “Identity-Based (Lossy) Trapdoor Functions and Applications”. In: *Advances in Cryptology, EUROCRYPT 2012*. Vol. 7237. Lecture Notes in Computer Science. Springer, 2012, pp. 228–245. URL: <https://eprint.iacr.org/2011/479>.
- [BKW17] Dan Boneh, Sam Kim, and David J. Wu. “Constrained Keys for Invertible Pseudorandom Functions”. In: *Theory of Cryptography Conference*. Vol. 10677. Lecture Notes in Computer Science. Springer, 2017, pp. 237–263.
- [BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. “On the Cryptographic Hardness of Finding a Nash Equilibrium”. In: *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*. Ed. by Venkatesan Guruswami. IEEE Computer Society, 2015, pp. 1480–1498.
- [BPW16] Nir Bitansky, Omer Paneth, and Daniel Wichs. “Perfect Structure on the Edge of Chaos: Trapdoor Permutations from Indistinguishability Obfuscation”. In: *Theory of Cryptography Conference*. Vol. 9562. Lecture Notes in Computer Science. Springer, 2016, pp. 474–502.
- [CL18] Ran Canetti and Amit Lichtenberg. “Certifying Trapdoor Permutations, Revisited”. In: *Theory of Cryptography Conference*. Vol. 11239. Lecture Notes in Computer Science. Springer, 2018, pp. 476–506.
- [CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. “Universally composable two-party and multi-party secure computation”. In: (2002), pp. 494–503.
- [CLZ26] Ran Canetti, Ji Luo, and Yiding Zhang. *How to Encrypt with Random Reversible Circuits*. Cryptology ePrint Archive, Paper 2026/1398. 2026. URL: <https://eprint.iacr.org/2026/1398>.
- [DH76] W. Diffie and M.E. Hellman. “New directions in cryptography”. In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654.
- [FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. “Multiple NonInteractive Zero Knowledge Proofs Under General Assumptions”. In: *SIAM Journal on Computing* 29.1 (1999), pp. 1–28.
- [GM84] Shafi Goldwasser and Silvio Micali. “Probabilistic encryption”. In: *Journal of Computer and System Sciences* 28.2 (1984), pp. 270–299.

- [GPSZ17] Sanjam Garg, Omkant Pandey, Akshayaram Srinivasan, and Mark Zhandry. “Breaking the Sub-Exponential Barrier in Obfustopia”. In: *Advances in Cryptology—EUROCRYPT 2017*. Vol. 10212. Lecture Notes in Computer Science. Springer, 2017, pp. 156–181.
- [GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. “Trapdoors for Hard Lattices and New Cryptographic Constructions”. In: *40th Annual ACM Symposium on Theory of Computing*. ACM, 2008, pp. 197–206. DOI: 10.1145/1374376.1374407.
- [GR13] Oded Goldreich and Ron D. Rothblum. “Enhancements of Trapdoor Permutations”. In: *Journal of Cryptology* 26.3 (2013), pp. 484–512.
- [HMQU18] Dennis Hofheinz, Jörn Müller-Quade, and Dominique Unruh. “On the (Im-)Possibility of Extending Coin Toss”. In: *Journal of Cryptology* 31.4 (2018), pp. 1126–1163. DOI: 10.1007/s00145-018-9296-x.
- [HS20] Martha Norberg Hovd and Martijn Stam. *Vetted Encryption*. Cryptology ePrint Archive, Paper 2020/1348. 2020. URL: <https://eprint.iacr.org/2020/1348>.
- [HY20] Pavel Hubáček and Eylon Yogev. “Hardness of Continuous Local Search: Query Complexity and Cryptographic Lower Bounds”. In: *SIAM Journal on Computing* 49.6 (2020), pp. 1128–1172. DOI: 10.1137/17M118014.
- [HYZLXY19] Meijuan Huang, Bo Yang, Yi Zhao, Kaitai Liang, Liang Xue, and Xiaoyi Yang. “CCA-Secure Deterministic Identity-Based Encryption Scheme”. In: *Journal of Universal Computer Science* 25.3 (2019), pp. 245–269. DOI: 10.3217/jucs-025-03-0245. URL: https://www.jucs.org/jucs_25_3/cca_secure_deterministic_identity/jucs_25_03_0245_0269_huang.pdf.
- [Hai04] Iftach Haitner. “Implementing Oblivious Transfer Using Collection of Dense Trapdoor Permutations”. In: *Theory of Cryptography Conference*. Vol. 2951. Lecture Notes in Computer Science. Springer, 2004, pp. 394–409. DOI: 10.1007/978-3-540-24638-1_22.
- [Hof12] Dennis Hofheinz. “All-But-Many Lossy Trapdoor Functions”. In: *Advances in Cryptology, EUROCRYPT 2012*. Vol. 7237. Lecture Notes in Computer Science. Springer, 2012, pp. 209–227. DOI: 10.1007/978-3-642-29011-4_14.
- [LQ05] Benoît Libert and Jean-Jacques Quisquater. “Identity Based Encryption Without Redundancy”. In: *Applied Cryptography and Network Security, ACNS 2005*. Vol. 3531. Lecture Notes in Computer Science. Springer, 2005, pp. 285–300. DOI: 10.1007/11496137_20.
- [MY23] Tomoyuki Morimae and Takashi Yamakawa. “Proofs of Quantumness from Trapdoor Permutations”. In: *14th Innovations in Theoretical Computer Science Conference (ITCS 2023)*. Vol. 251. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 87:1–87:14. DOI: 10.4230/LIPIcs.ITCS.2023.87.
- [Nao91] Moni Naor. “Bit Commitment Using Pseudorandomness”. In: *Journal of Cryptology* 4.2 (1991). Preliminary version appeared in CRYPTO ’89., pp. 151–158.
- [PW08] Chris Peikert and Brent Waters. “Lossy Trapdoor Functions and Their Applications”. In: *40th Annual ACM Symposium on Theory of Computing*. ACM, 2008, pp. 187–196. DOI: 10.1145/1374376.1374406.
- [Pap94] Christos H. Papadimitriou. “On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence”. In: *J. Comput. Syst. Sci.* 48.3 (1994), pp. 498–532.
- [Pie90] Josef Pieprzyk. “How to Construct Pseudorandom Permutations from Single Pseudorandom Functions”. In: *Advances in Cryptology - EUROCRYPT ’90, Workshop on the Theory and Application of Cryptographic Techniques, Aarhus, Denmark, May 21-24, 1990, Proceedings*. Ed. by Ivan Damgård. Lecture Notes in Computer Science. Springer, 1990, pp. 140–150.

- [SP92] Alfredo De Santis and Giuseppe Persiano. “Zero-Knowledge Proofs of Knowledge Without Interaction (Extended Abstract)”. In: *33rd Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 1992, pp. 427–436. DOI: 10.1109/SFCS.1992.267809.
- [SW14] Amit Sahai and Brent Waters. “How to Use Indistinguishability Obfuscation: Deniable Encryption, and More”. In: *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*. ACM, 2014, pp. 475–484. DOI: 10.1145/2591796.2591825. URL: <https://eprint.iacr.org/2013/454>.
- [SZ25] Omri Shmueli and Mark Zhandry. “On One-Shot Signatures, Quantum vs Classical Binding, and Obfuscating Permutations”. In: *IACR Cryptol. ePrint Arch.* (2025), p. 486. URL: <https://eprint.iacr.org/2025/486>.
- [XXZ12] Xiang Xie, Rui Xue, and Rui Zhang. *Deterministic Public Key Encryption and Identity-Based Encryption from Lattices in the Auxiliary-Input Setting*. Cryptology ePrint Archive, Paper 2012/463. 2012. URL: <https://eprint.iacr.org/2012/463>.
- [Yao82] Andrew Chi-Chih Yao. “Theory and Applications of Trapdoor Functions (Extended Abstract)”. In: *23rd Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 1982, pp. 80–91.

A Generalization of the sparse trigger lemma

We generalize the sparse random trigger lemma of Shmueli and Zhandry [SZ25, Lemma 17], which establishes indistinguishability and unpredictability for a uniformly sampled trigger. Here, the trigger is sampled from a hard distribution as in Definition 3.4, and the guarantees hold jointly for ℓ circuits sharing the same trigger. We give explicit bounds in terms of the source hardness and the obfuscation security error.

Notation. For every $y^*, y' \in [N]$, define the transposition $\Gamma^{(y^*, y')} : [N] \rightarrow [N]$ by

$$\Gamma^{(y^*, y')}(y) := \begin{cases} y^* & \text{if } y = y', \\ y' & \text{if } y = y^*, \\ y & \text{otherwise.} \end{cases}.$$

Sparse random triggers. Let C and C' be circuits with the same input and output domains, let R be a function with range $[N]$, and let $y \in [N]$. Define the joint program $J_{R=y}[C, C']$ by

$$J_{R=y}[C, C'](x) := \begin{cases} C(x) & \text{if } R(x) \neq y, \\ C'(x) & \text{if } R(x) = y. \end{cases}.$$

Lemma A.1 (Source sparse-trigger). *Let $N = N(\lambda)$ be an efficiently computable domain-size function that is exponential in λ . Let $\mathcal{D} = \{\mathcal{D}_N\}_{N \in \mathbb{N}}$ be a $\epsilon_{\mathcal{D}}$ -hard collection of distributions over $[N]$ as defined in Definition 3.4.*

Assume the existence of an indistinguishability obfuscator iO with security error $\epsilon_{\text{iO}}(\cdot)$. Then, the following two properties hold:

- **Unpredictability.** *For every $\lambda, \ell \in \mathbb{N}$, let $s = \text{poly}(\lambda, \log N)$. For any two tuples of ℓ circuits $(C_1, \dots, C_\ell)$ and $(C'_1, \dots, C'_\ell)$, any tuple of ℓ circuits $(R_1, \dots, R_\ell)$ with range $[N]$, and every PPT*

adversary $\mathbf{A}$,

$$\Pr \left[\mathbf{A} \left(C_1, \dots, C_\ell, \widehat{C}_1, \dots, \widehat{C}_\ell \right) = y \left| \begin{array}{l} y \leftarrow \mathcal{D}_N \\ \widehat{C}_1 \leftarrow \text{iO}(1^\lambda, 1^s, J_{R_1=y}[C_1, C'_1]) \\ \dots \\ \widehat{C}_\ell \leftarrow \text{iO}(1^\lambda, 1^s, J_{R_\ell=y}[C_\ell, C'_\ell]) \end{array} \right. \right] \leq \frac{1}{N} + \epsilon_{\mathcal{D}}(N) + \ell \epsilon_{\text{iO}}(\lambda) .$$

- **Indistinguishability.** For every $\lambda, \ell \in \mathbb{N}$, let $s = \text{poly}(\lambda, \log N)$. For any two tuples of ℓ circuits $(C_1, \dots, C_\ell)$ and $(C'_1, \dots, C'_\ell)$, and any tuple of ℓ circuits $(R_1, \dots, R_\ell)$ with range $[N]$, the obfuscations of $(C_1, \dots, C_\ell)$ are indistinguishable from the obfuscations of $(J_{R_1=y}[C_1, C'_1], \dots, J_{R_\ell=y}[C_\ell, C'_\ell])$, where $y \leftarrow \mathcal{D}_N$. Formally,

$$\left\{ (\widehat{C}_1, \dots, \widehat{C}_\ell) \left| \begin{array}{l} \widehat{C}_1 \leftarrow \text{iO}(1^\lambda, 1^s, C_1) \\ \dots \\ \widehat{C}_\ell \leftarrow \text{iO}(1^\lambda, 1^s, C_\ell) \end{array} \right. \right\} \approx_{\frac{1}{N} + \epsilon_{\mathcal{D}}(N) + 2\ell \epsilon_{\text{iO}}(\lambda)} \left\{ (\widehat{C}_1, \dots, \widehat{C}_\ell) \left| \begin{array}{l} y \leftarrow \mathcal{D}_N \\ \widehat{C}_1 \leftarrow \text{iO}(1^\lambda, 1^s, J_{R_1=y}[C_1, C'_1]) \\ \dots \\ \widehat{C}_\ell \leftarrow \text{iO}(1^\lambda, 1^s, J_{R_\ell=y}[C_\ell, C'_\ell]) \end{array} \right. \right\} .$$

Proof of Lemma A.1. We adapt the injective-PRG argument of [SZ25, proof of Lemma 17], using the hard-source guarantee from Definition 3.4 in place of pseudorandomness on a uniform seed, and applying the iO steps to each of the ℓ circuits. We first prove unpredictability and then indistinguishability.

Unpredictability. Let $\mathbf{G}_{\mathcal{D}} := \{\mathbf{G}_N\}_{N \in \mathbb{N}}$ be a collection of length-doubling injective circuits as defined in Definition 3.4 relative to the $\epsilon_{\mathcal{D}}$ -hard collection $\mathcal{D}$. Fix λ, N, ℓ, s , and the circuit tuples as quantified in the unpredictability property. For shorthand, set $\mathbf{G} := \mathbf{G}_N$ and $\mathcal{D} := \mathcal{D}_N$. Fix some PPT adversary $\mathbf{A}$, and let ϵ be the probability that it predicts y . Formally,

$$\epsilon := \Pr \left[\mathbf{A} \left(C_1, \dots, C_\ell, \widehat{C}_1, \dots, \widehat{C}_\ell \right) = y \left| \begin{array}{l} y \leftarrow \mathcal{D} \\ \widehat{C}_1 \leftarrow \text{iO}(1^\lambda, 1^s, J_{R_1=y}[C_1, C'_1]) \\ \dots \\ \widehat{C}_\ell \leftarrow \text{iO}(1^\lambda, 1^s, J_{R_\ell=y}[C_\ell, C'_\ell]) \end{array} \right. \right] .$$

Since $\mathbf{G}$ is injective, checking whether $\mathbf{G}(R_i(x)) = \mathbf{G}(y)$ is equivalent to checking whether $R_i(x) = y$. Therefore, by iO security,

$$\Pr \left[\mathbf{A} \left(C_1, \dots, C_\ell, \widehat{C}_1, \dots, \widehat{C}_\ell \right) = y \left| \begin{array}{l} y \leftarrow \mathcal{D} \\ w \leftarrow \mathbf{G}(y) \\ \widehat{C}_1 \leftarrow \text{iO}(1^\lambda, 1^s, J'_{R_1=w}[C_1, C'_1]) \\ \dots \\ \widehat{C}_\ell \leftarrow \text{iO}(1^\lambda, 1^s, J'_{R_\ell=w}[C_\ell, C'_\ell]) \end{array} \right. \right] \geq \epsilon - \ell \cdot \epsilon_{\text{iO}}(\lambda) . \quad (26)$$

Observe that the view of the adversary in the above experiment depends on y only through $w = \mathbf{G}(y)$. We construct the following PPT distinguisher $\mathbf{D}$ for the hardness of $\mathcal{D}$:

$\mathbf{D}(w)$:

1. For every $i \in [\ell]$, compute $\hat{C}_i \leftarrow \text{iO}(1^\lambda, 1^s, J'_{R_i=w}[C_i, C'_i])$.
2. Emulate $\hat{y} \leftarrow \mathbf{A}(C_1, \dots, C_\ell, \hat{C}_1, \dots, \hat{C}_\ell)$.
3. If $\hat{y} \in [N]$ and $\mathbf{G}(\hat{y}) = w$, then output 1
4. Else, output 0.

We analyze the probability that the distinguisher outputs 1 in the following two cases:

- $y \leftarrow \mathcal{D}$ and $w \leftarrow \mathbf{G}(y)$: In this case, $\mathbf{D}$ generates exactly the obfuscations in the preceding experiment. Moreover, injectivity of $\mathbf{G}$ gives

$$\mathbf{G}(\hat{y}) = w = \mathbf{G}(y) \iff \hat{y} = y.$$

Thus, the acceptance probability of $\mathbf{D}$ in the hard-distribution experiment is exactly the prediction probability in Equation 26. That is,

$$\Pr \left[\mathbf{D}(w) = 1 \mid \begin{array}{l} y \leftarrow \mathcal{D} \\ w \leftarrow \mathbf{G}(y) \end{array} \right] \geq \epsilon - \ell \cdot \epsilon_{\text{iO}}(\lambda) .$$

- $w \leftarrow [N^2]$: In this case, if $\mathbf{D}$ outputs 1, then $w = \mathbf{G}(\hat{y})$ for some $\hat{y} \in [N]$, and hence $w \in \text{Img}(\mathbf{G})$. Notice that in this case, we do not sample a value from $\mathcal{D}$, and $\hat{y}$ may be any point in $[N]$. Thus the relevant set is the full image of $\mathbf{G}$. Since $\mathbf{G}$ is injective,

$$|\text{Img}(\mathbf{G})| = N.$$

Therefore,

$$\begin{aligned} \Pr_{w \leftarrow [N^2]}[\mathbf{D}(w) = 1] &\leq \Pr_{w \leftarrow [N^2]}[w \in \text{Img}(\mathbf{G})] \\ &= \frac{|\text{Img}(\mathbf{G})|}{N^2} = \frac{1}{N}. \end{aligned}$$

The $\epsilon_{\mathcal{D}}$ -hardness of $\mathcal{D}$ according to Definition 3.4 therefore implies

$$\Pr \left[\mathbf{A}(C_1, \dots, C_\ell, \hat{C}_1, \dots, \hat{C}_\ell) = y \mid \begin{array}{l} y \leftarrow \mathcal{D} \\ w \leftarrow \mathbf{G}(y) \\ \hat{C}_1 \leftarrow \text{iO}(1^\lambda, 1^s, J'_{R_1=w}[C_1, C'_1]) \\ \dots \\ \hat{C}_\ell \leftarrow \text{iO}(1^\lambda, 1^s, J'_{R_\ell=w}[C_\ell, C'_\ell]) \end{array} \right] \leq \frac{1}{N} + \epsilon_{\mathcal{D}}(N) ,$$

Overall, the adversary predicts y with probability

$$\epsilon \leq \frac{1}{N} + \epsilon_{\mathcal{D}}(N) + \ell \cdot \epsilon_{\text{iO}}(\lambda) ,$$

as required.

Indistinguishability. Let $\mathbf{G}_{\mathcal{D}} := \{\mathbf{G}_N\}_{N \in \mathbb{N}}$ be a collection of length-doubling injective circuits as defined in Definition 3.4 relative to the $\epsilon_{\mathcal{D}}$ -hard collection $\mathcal{D}$. Fix λ, N, ℓ, s , and the circuit tuples as quantified in the indistinguishability property. For shorthand, set $\mathbf{G} := \mathbf{G}_N$ and $\mathcal{D} := \mathcal{D}_N$. We prove indistinguishability using the following sequence of hybrids.

- Hybrid 0: In this hybrid we obfuscate the original circuits $(C_1, \dots, C_\ell)$:

$$\left\{ (\hat{C}_1, \dots, \hat{C}_\ell) \left| \begin{array}{l} \hat{C}_1 \leftarrow \text{iO}(1^\lambda, 1^s, C_1) \\ \dots \\ \hat{C}_\ell \leftarrow \text{iO}(1^\lambda, 1^s, C_\ell) \end{array} \right. \right\}.$$

- Hybrid 1: In this hybrid, for every $i \in [\ell]$, we obfuscate $J'_{R_i=w}[C_i, C'_i]$, where $w \leftarrow [N^2]$, and $J'_{R_i=w}[C_i, C'_i]$ is defined as follows:

$$J'_{R_i=w}[C_i, C'_i](x) := \begin{cases} C_i(x) & \text{if } \mathbf{G}(R_i(x)) \neq w, \\ C'_i(x) & \text{if } \mathbf{G}(R_i(x)) = w. \end{cases}$$

Since

$$\Pr_{w \leftarrow [N^2]}[w \in \text{Im}(\mathbf{G})] \leq \frac{1}{N},$$

with probability at least $1 - 1/N$ the second line is not triggered in any of the obfuscated circuits. Therefore, $J'_{R_i=w}[C_i, C'_i]$ has the same functionality as C_i . In this case, $\text{iO}(1^\lambda, 1^s, J'_{R_i=w}[C_i, C'_i])$ is $\epsilon_{\text{iO}}(\lambda)$ -indistinguishable from $\text{iO}(1^\lambda, 1^s, C_i)$. Overall, we get that

$$\begin{aligned} & \left\{ (\hat{C}_1, \dots, \hat{C}_\ell) \left| \begin{array}{l} \hat{C}_1 \leftarrow \text{iO}(1^\lambda, 1^s, C_1) \\ \dots \\ \hat{C}_\ell \leftarrow \text{iO}(1^\lambda, 1^s, C_\ell) \end{array} \right. \right\} \\ & \approx_{\frac{1}{N} + \ell \epsilon_{\text{iO}}(\lambda)} \left\{ (\hat{C}_1, \dots, \hat{C}_\ell) \left| \begin{array}{l} w \leftarrow [N^2] \\ \hat{C}_1 \leftarrow \text{iO}(1^\lambda, 1^s, J'_{R_1=w}[C_1, C'_1]) \\ \dots \\ \hat{C}_\ell \leftarrow \text{iO}(1^\lambda, 1^s, J'_{R_\ell=w}[C_\ell, C'_\ell]) \end{array} \right. \right\}. \end{aligned}$$

- Hybrid 2: In this hybrid, for every $i \in [\ell]$, we obfuscate $J'_{R_i=w}[C_i, C'_i]$, where w is sampled from the image of $\mathbf{G}$: $y \leftarrow \mathcal{D}$ and $w \leftarrow \mathbf{G}(y)$. The following indistinguishability follows from the $\epsilon_{\mathcal{D}}$ -hardness of $\mathcal{D}$:

$$\begin{aligned} & \left\{ (\hat{C}_1, \dots, \hat{C}_\ell) \left| \begin{array}{l} w \leftarrow [N^2] \\ \hat{C}_1 \leftarrow \text{iO}(1^\lambda, 1^s, J'_{R_1=w}[C_1, C'_1]) \\ \dots \\ \hat{C}_\ell \leftarrow \text{iO}(1^\lambda, 1^s, J'_{R_\ell=w}[C_\ell, C'_\ell]) \end{array} \right. \right\} \\ & \approx_{\epsilon_{\mathcal{D}}(N)} \left\{ (\hat{C}_1, \dots, \hat{C}_\ell) \left| \begin{array}{l} y \leftarrow \mathcal{D} \\ w \leftarrow \mathbf{G}(y) \\ \hat{C}_1 \leftarrow \text{iO}(1^\lambda, 1^s, J'_{R_1=w}[C_1, C'_1]) \\ \dots \\ \hat{C}_\ell \leftarrow \text{iO}(1^\lambda, 1^s, J'_{R_\ell=w}[C_\ell, C'_\ell]) \end{array} \right. \right\}. \end{aligned}$$

- Hybrid 3: In this hybrid, for every $i \in [\ell]$, we obfuscate $J_{R_i=y}[C_i, C'_i]$, where $y \leftarrow \mathcal{D}$. Recall that

$$J_{R_i=y}[C_i, C'_i](x) := \begin{cases} C_i(x) & \text{if } R_i(x) \neq y, \\ C'_i(x) & \text{if } R_i(x) = y. \end{cases}$$

Since $\mathbf{G}$ is injective, checking whether $\mathbf{G}(R_i(x)) = \mathbf{G}(y)$ is equivalent to checking whether $R_i(x) = y$. Therefore, by $\mathbf{iO}$ security, Hybrid 2 and Hybrid 3 are $\ell \cdot \epsilon_{\mathbf{iO}}(\lambda)$ -indistinguishable.

Overall, we get that

$$\left\{ (\hat{C}_1, \dots, \hat{C}_\ell) \left| \begin{array}{l} \hat{C}_1 \leftarrow \mathbf{iO}(1^\lambda, 1^s, C_1) \\ \dots \\ \hat{C}_\ell \leftarrow \mathbf{iO}(1^\lambda, 1^s, C_\ell) \end{array} \right. \right\} \\ \approx_{\frac{1}{N} + \epsilon_{\mathcal{D}}(N) + 2\ell\epsilon_{\mathbf{iO}}(\lambda)} \left\{ (\hat{C}_1, \dots, \hat{C}_\ell) \left| \begin{array}{l} y \leftarrow \mathcal{D} \\ \hat{C}_1 \leftarrow \mathbf{iO}(1^\lambda, 1^s, J_{R_1=y}[C_1, C'_1]) \\ \dots \\ \hat{C}_\ell \leftarrow \mathbf{iO}(1^\lambda, 1^s, J_{R_\ell=y}[C_\ell, C'_\ell]) \end{array} \right. \right\}.$$

□

Acknowledgments

Shany Ben-David is supported by the European Research Union (ERC, CRYPTOPROOF, 101164375), and by the Clore Israel Foundation. Eylon Yogev is supported by the Israel Science Foundation (Grant No. 2302/22) and the European Research Union (ERC, CRYPTOPROOF, 101164375). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.