How to Encrypt with Random Reversible Circuits

Ran Canetti (✉) 罗辑 (Ji Luo) (✉) Yiding Zhang (✉)

Boston University

canetti@bu.edu luoji@cs.washington.edu zyding@bu.edu

8 July 2026

Abstract

This work revisits a natural paradigm for constructing public-key encryption, whereby the public key is an obfuscated block cipher in encryption mode. We show that if the block cipher is a permutable pseudorandom permutation [Shmueli–Zhandry, Crypto '25] and the obfuscator is indistinguishability-secure, then the following holds.

1. Applying the obfuscated cipher directly to the message and a short random nonce, without any additional structure or consistency checks, suffices for CCA2 security.
2. Augmenting the scheme with the capability to generate obfuscated decrypt-then-apply-f circuits (for any given function f), yields a *functional encryption* scheme that is *simulation-secure against adaptive chosen-ciphertext attacks*.
3. For any length-preserving function g, augmenting the public key with an obfuscated decrypt-apply-g-reencrypt circuit allows anyone to homomorphically apply g to encrypted data, for an unbounded number of times, while preserving semantic security. (This relies on subexponential security.)

We also show that, under the split-circuit pseudorandomness (SCP) assumption of [Canetti–Chamon–Mucciolo–Ruckenstein, TCC '24], random reversible circuits form a permutable pseudorandom permutation family. This points to obfuscated random reversible circuits as a potential alternative avenue to public-key encryption with strong security and rich functionality.

This is the full version of the first part of

How to Encrypt with Random Reversible Circuits: Functional, Homomorphic and CCA-Secure [CLZ26a] in the proceedings of Crypto 2026 to be published by Springer.

The full version of the second part is [CLZ26b].

Contents

1 Introduction **1**
- 1.1 This Work . . . 1

2 Technical Overview **3**
- 2.1 Straight-Forward Public-Key Encryption . . . 4
- 2.2 Free Upgrade to Functional Encryption . . . 7
- 2.3 Optimized Key Encapsulation Mechanism . . . 7
- 2.4 Unary Homomorphic Encryption . . . 8
- 2.5 Instantiation with Reversible Circuits . . . 11

3 Preliminaries **12**

4 Functional Encryption with SIM-CCA Security (+ PKE) **15**

5 Key Encapsulation Mechanism **24**

6 Unary Homomorphic Encryption **28**
- 6.1 Lemma(s) . . . 30
- 6.2 Security Proof . . . 32
- 6.3 Barrier to Richer Homomorphisms (Collision Resistance) . . . 36

7 New Notions for Trapdoor/Pseudorandom Permutations **38**
- 7.1 Puncturable TDP with Multiple Modes of Puncturing . . . 38
- 7.2 Puncturable TDP with Cohardcore Functions for Distorted Input . . . 42
- 7.3 Puncturable PRP . . . 47

8 Instantiations with Reversible Circuits **48**
- 8.1 Section Preliminaries . . . 48
- 8.2 Permutable PRP . . . 50
- 8.3 Reversible-Circuit Trigger . . . 52
- 8.4 Useful Gadgets of Reversible Circuits . . . 56
- 8.5 KEM and (Plaintext-Selective) PKE with CCA Security . . . 59

References **63**

Appendix

A Proof of Lemma 2 (Random Triggers) **67**

1 Introduction

Public-key encryption (PKE), as originally proposed by Diffie and Hellman [DH76] half a century ago, envisions the public encryption algorithm as the result of a "one-way transformation" applied to a block cipher in encryption mode. In today's terminology, this translates to the public key being an obfuscated version of a pseudorandom permutation.

To be a bit more concrete, consider the following natural, if somewhat simplistic symmetric encryption scheme for n-bit messages, using a block cipher π on $(n + \lambda)$-bit blocks. To encrypt message x, choose a random λ-bit nonce r and set the ciphertext to be $c \leftarrow \pi(x\|r)$. To decrypt c, output the first n bits of $\pi^{-1}(c)$. It is easy to see that this scheme is semantically secure, with λ being the security parameter. In fact, it is even secure against adaptive chosen-ciphertext attacks, as-is, with no additional structure or consistency checks. Diffie and Hellman's vision points to designing public key encryption by way of obfuscating the above encryption process.

The actual public-key encryption schemes developed over the past five decades do not quite follow this simple template. Starting from the Diffie–Hellman key exchange, followed by the RSA permutation [RSA78], the Goldwasser–Micali encryption scheme [GM82], and a great many others, we have invariably found alternative ways to obtain the asymmetry needed for guaranteeing security in the presence of a publicly available encryption process — often at significant cost in both security and efficiency. Even public-key encryption schemes that directly obfuscate a symmetric encryption process, such as that of Sahai–Waters [SW14], use costly additional constructs to work around the inability to directly apply indistinguishability obfuscation to block ciphers.

1.1 This Work

This work addresses the challenge of making Diffie and Hellman's simplistic idea work. Specifically, we employ the Shmueli–Zhandry [SZ25] notion of permutable pseudorandom permutations (PPRP), which can be thought of as a natural extension of puncturable pseudorandom functions to permutations. In a nutshell, we show that if the underlying block cipher π is a PPRP, then setting the public key to $i\mathcal{O}(\pi)$, where $i\mathcal{O}$ is an indistinguishability obfuscator, turns the above encryption scheme into a CCA2-secure PKE. We then extend this result in a number of ways.

SIM-CCA Functional Encryption. We show that this same PKE scheme can be naturally augmented to achieve functional encryption. For a circuit $C : \{0,1\}^n \to \{0,1\}^\star$, its functional key is an obfuscated version of decrypt-then-apply-C, i.e., $\pi(x\|r) \mapsto C(x)$, with *zero* integrity verification logic inside. The scheme is *simulation*-secure for semi-adaptive x against *adaptive* chosen-ciphertext attacks.[1] (We also propose this notion, termed SIM-CCA, not having found it in the literature.) It is the first[2] FE scheme for general circuits secure in the presence of adaptive access to the decryption oracle.

[1] We remark that existing FE schemes where functional keys are obfuscated circuits, such as [GGH+13, Wat15,GWZ22], do perform cryptographic checks (NIZK, signature, etc.) under obfuscation, yet only enjoy indistinguishability-based security.

[2] The security proof for a previous construction [PD21a] contains an error, unfixed as of the writing of this work (see [PD21b] for details). The notion considered there is x-adaptive IND-CCA, which is incomparable to SIM-CCA (see Remark 1) and remains open.

Unary Homomorphic Encryption. Orthogonally, we show that the same simple scheme maintains semantic security when augmented with a homomorphic evaluation program for any circuit $g : \{0,1\}^n \to \{0,1\}^n$, again in a natural manner. The program is simply an obfuscated version of decrypt-apply-g-reencrypt, $\pi(x\|r) \mapsto \pi(g(x)\|\sigma'(r))$, where σ' is for generating new encryption randomness. This enables everyone to homomorphically apply g on any *single* ciphertext. Following [PR08], we call such homomorphic encryption schemes *unary*.

It is stressed that g operates on all n bits of the plaintext. Furthermore, the scheme is unleveled, i.e., it supports iterated homomorphic evaluations of g with no *a priori* bound on the number of iterations and the ciphertext remains $(n + \mathrm{poly}(\lambda))$-bit long throughout, with n being an arbitrarily large polynomial in λ. However, we can only demonstrate semantic security with *one* homomorphic evaluation program per instance of the scheme.

(Barrier to) Multi-Function Unary HE. The capability of iterating a single unary function already covers many natural use cases, including the original motivation of enabling user-specified remote processing of encrypted data [RAD78]. It is nevertheless an attractive generalization to support multiple unary evaluation programs, each for a different function, with respect to the same decryption key. While we do not have a concrete reason to believe that such an extension is necessarily insecure, there is evidence that existing proof techniques are insufficient. (See more details within.) We leave further analysis of this extension as an intriguing open problem.

Optimized Key Encapsulation Mechanism. So far, we concentrated on having the plaintext message as a direct input to the obfuscated PRP. In addition, we show that setting $\mathsf{ct} \leftarrow \pi(r)$ to encapsulate $k \leftarrow \sigma(r)$ (where r is sampled by the encapsulator and σ is another PPRP) results in a CCA-secure key encapsulation mechanism (KEM). This means that k can be used in full as a symmetric encryption key, while maintaining optimal ciphertext size (namely, $|\mathsf{ct}| = |k|$ with k having $|k|$ bits of entropy).

Implementation via Mixed Reversible Circuits. While the above results may be conceptually pleasing and elegant, so far they arguably are of purely theoretical interest. Indeed, the Shmueli–Zhandry construction of PPRP involves obfuscating general one-way functions using general-purpose indistinguishability obfuscation ($i\mathcal{O}$), which renders it prohibitively complex.

We thus complement our results by demonstrating an alternative route to PPRP. This route is based on the potential to obtain general-purpose obfuscators by way of locally mixing reversible circuits, as proposed in [CCMR24]. Such reversible-circuit obfuscators (rc$i\mathcal{O}$) preserve the format of reversible circuits, i.e., the obfuscation of a reversible circuit is a functionally equivalent one with the same number of wires (and more gates).

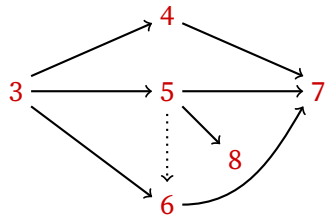
Specifically, we show that under the *split-circuit pseudorandomness* (SCP) assumption (used in [CCMR24] to construct rc$i\mathcal{O}$), a *random* n-wire m-gate reversible circuit π is a permutable PRP for some $n, m \in \mathrm{poly}(\lambda)$. Furthermore, we prove that applying rc$i\mathcal{O}$ to the Toffoli version of π, i.e., $C(x, y) := (x, y \oplus \pi(x))$ without additional ancillae, yields a trapdoor permutation, as well as PKE and KEM.

New Security Notions for TDP and PRP. Along the way of obtaining the various advanced encryption schemes, we find a few simple-to-state properties of TDP and

PRP satisfied by obfuscated PPRP: multi-mode puncturable TDP (in-)sufficient for CCA-secure PKE; TDP with pseudorandom images for distorted input distributions against chosen-ciphertext attacks; puncturable PRP. We formulate those notions and hope that further research leads to simpler constructions of them.

Composition of Functional Decryption and Homomorphic Evaluation. Given the results exhibited so far, one wonders what security properties can be guaranteed if the same PKE is augmented with *both* functional keys *and* homomorphic evaluation programs. Does it retain *any* security, or wishfully, strong security against malleability attacks? If so, how might one capture such security properties? Furthermore, why should the composed capabilities be of interest? This turns to be an intriguing set of questions, which are the focus of a companion paper [CLZ26b].

Organization. Section 2 is the technical overview, in the same structure as the main text. The preliminaries used throughout the paper are in Section 3. In Section 4, we present the definition and construction of FE with SIM-CCA security, which subsumes adaptive CCA-secure PKE. In Section 5, we present the size-optimized KEM. In Section 6, we present our unary HE. Section 7 abstracts new notions of trapdoor/pseudorandom permutations arising from Sections 4, 5, and 6, and includes some further discussions. In Section 8, we show how to instantiate permutable PRP, CCA-secure KEM, and (plaintext-selective) CCA-secure PKE using reversible circuits. The dependencies among the sections are as follows.



A solid arrow means conceptual dependency and a dotted arrow means technical-only dependency (Section 6 uses a lemma proven in Section 5).

2 Technical Overview

We start with an *operational characterization* of permutable PRP [SZ25]. Let $i\mathcal{O}$ be an indistinguishability obfuscator. An (input-)permutable PRP family is a distribution over pairs of programs (π, π^{-1}) that compute a permutation over some domain (say, $\{0,1\}^N$) and its inverse, and the distribution satisfies

$$\left(i\mathcal{O}(\pi),\, i\mathcal{O}(\pi^{-1})\right) \approx \left(i\mathcal{O}(\pi \circ \Gamma),\, i\mathcal{O}(\Gamma^{-1} \circ \pi^{-1})\right)$$

for any adversarially chosen permutation Γ (from a set of allowed ones) and randomly sampled (π, π^{-1}).

Shmueli and Zhandry show that if a PPRP family is permutable with respect to all transpositions, then $i\mathcal{O}(\pi)$ is a trapdoor one-way permutation with π^{-1} being the trapdoor [SZ25],[3] when $\lambda \le N \le \mathrm{poly}(\lambda)$. It is illuminating to recall their elegant proof.

[3] Their original proof uses *output*-permutable PRP, where Γ (resp. Γ^{-1}) is applied after π (resp. before π^{-1}). There is no difference in their feasibility. We find input-permutability cleaner for our expositions.

Consider an adversary that, given $i\mathcal{O}(P)$ (with $P = \pi$) and $y^* = \pi(x^*)$ for a random x^*, outputs x^* with non-negligible probability, then... *i)* plant in P a random trigger so that another random point x' collide with x^*, i.e., make $P(x') = \pi(x^*)$ and $P(x) = \pi(x)$ for $x \neq x'$; *ii)* invoke PPRP security to swap the roles of x' and x^* within the PPRP itself, i.e., permute π by $\Gamma := (x' \;\; x^*)$, which makes $P(x^*) = \pi(x')$ (and $P(x) = \pi(x)$ for $x \neq x^*$) and $y^* = \pi(x')$. After the two steps, x^* is a random trigger — P can be written using any code that decides whether $x = x^*$ and so can the winning condition — so *iii)* remove x^* as a random trigger, which simultaneously reverts P to π and makes the winning condition always fail. But all the three transitions are indistinguishable, leading to a contradiction, hence proving one-wayness.

As we shall see, the trigger–permute–untrigger technique can be further exploited to achieve much more than TDP.

2.1 Straight-Forward Public-Key Encryption

The mere fact that $\widetilde{\pi} \xleftarrow{\$} i\mathcal{O}(\pi)$ is a TDP does not imply, of course, the security of the simplistic PKE scheme defined by $\mathsf{Enc}(\widetilde{\pi}, x; r) := \widetilde{\pi}(x \| r)$. Indeed, the "textbook" way to build PKE from TDP is significantly less attractive. One needs to find a hardcore function h of the TDP and use them as a key encapsulation mechanism (KEM), i.e., $\mathsf{Enc}(\widetilde{\pi}, x; r) := (\widetilde{\pi}(r),\, h(r) \oplus x)$.

To prove the security of the simplistic scheme, we tweak the above proof to show that if π is a PPRP, then $\pi(x)$ is *pseudorandom* given $i\mathcal{O}(\pi)$ as soon as a sufficiently long substring of x is random, even if *the rest of x is fixed*. For any adversarially chosen $v^* \in \{0,1\}^\star$, let $r^* \xleftarrow{\$} \{0,1\}^\lambda$, then

$$\big(i\mathcal{O}(\pi),\, \pi(v^* \| r^*)\big) \approx \big(i\mathcal{O}(\pi),\, \text{random}\big), \tag{1}$$

which implies that the PKE scheme is *selectively* secure (more on this later). Denote by $P\{a \mapsto b\}$ the program P modified to map a to b and suppose that the adversary is always given $\mathsf{pk} \xleftarrow{\$} i\mathcal{O}(P)$ and ct^*. The proof proceeds as follows, with $x' \xleftarrow{\$} \{0,1\}^{|v^*|+\lambda}$:

$$\begin{aligned}
\big(P, \mathsf{ct}^*\big) &= \big(\pi,\, \pi(v^* \| r^*)\big) \\
\text{(add random trigger } x') \qquad &\approx \big(\pi\{x' \mapsto \pi(v^* \| r^*)\},\, \pi(v^* \| r^*)\big) \\
\text{(permute } \pi \text{ by } (x' \;\; v^* \| r^*)) \qquad &\approx \big(\pi\{v^* \| r^* \mapsto \pi(x')\},\, \pi(x')\big) \\
\text{(remove random trigger } r^*) \qquad &\approx \big(\pi,\, \pi(x')\big) \equiv \big(\pi,\, \text{random}\big).
\end{aligned}$$

Along the hybrids, $i\mathcal{O}$ steps to rewrite programs in equivalent ways are skipped to simplify exposition in this introduction. A key difference from the proof in [SZ25] is that only a portion of the input (corresponding to r^*) is used as a trigger in the last step.

Chosen-Ciphertext Security. CCA security for PKE is traditionally obtained by adding integrity checks to the ciphertext and during decryption. We show that the simplistic PKE scheme, with no additional structure or checks, is already CCA-secure. Note that the *secret*-key encryption scheme $\mathsf{Enc}(\pi, x; r) := \pi(x \| r)$ is CCA-secure when π is a plain strong PRP. Permutability is useful only in the upgrade to PKE via obfuscation.

Indeed, while any $(n + \lambda)$-bit string is a legitimate ciphertext with a well-defined n-bit plaintext, the ability to decrypt strings other than the challenge ct^* is of no help

to the adversary in the secret-key version — we show that this also holds for the public-key version. In fact, Equation (1) holds even in the presence of a *punctured decryption circuit* that decrypts every ciphertext other than ct^*. Let $P \setminus S$ be the program P modified to output $\perp$ on any input in S, then for any adversarially chosen v^*,

$$\big(i\mathcal{O}(\pi),\, y_0,\, i\mathcal{O}(\pi^{-1} \setminus \{y_0\})\big) \approx \big(i\mathcal{O}(\pi),\, y_1,\, i\mathcal{O}(\pi^{-1} \setminus \{y_1\})\big), \tag{2}$$

where $r^* \xleftarrow{\$} \{0,1\}^\lambda$, $y_0 \leftarrow \pi(v^* \| r^*)$, $y_1 \xleftarrow{\$} \{0,1\}^{|v^*|+\lambda}$.

In the previous proof, we have *underutilized* PPRP security by ignoring π^{-1}. The trick to prove Equation (2) is to trigger programs at *correlated* points. Suppose the adversary always receives $(i\mathcal{O}(P), y^*, i\mathcal{O}(Q))$. On the left-hand side, we have

$$P = \pi, \qquad y^* = \pi(v^* \| r^*), \qquad Q = \pi^{-1} \setminus \{y^*\} = \pi^{-1}\{\pi(v^* \| r^*) \mapsto \perp\}.$$

First, we plant a trigger $x' \xleftarrow{\$} \{0,1\}^{|v^*|+\lambda}$ into both programs:

$$\begin{aligned} P = \pi \quad &\big\{ \quad x' \mapsto \pi(v^* \| r^*), \qquad v^* \| r^* \mapsto \pi(v^* \| r^*) \big\}, \qquad y^* = \pi(v^* \| r^*), \\ Q = \pi^{-1} &\big\{ \pi(x') \mapsto v^* \| r^* \quad , \quad \pi(v^* \| r^*) \mapsto \perp \qquad \big\}. \end{aligned}$$

The redundant branch $v^* \| r^* \mapsto \cdots$ in P is for clarity. Permuting π by $(x' \;\; v^* \| r^*)$ leads to

$$\begin{aligned} P = \pi \quad &\big\{ \qquad x' \mapsto \pi(x'), \quad v^* \| r^* \mapsto \pi(x') \big\}, \qquad y^* = \pi(x'), \\ Q = \pi^{-1} &\big\{ \pi(v^* \| r^*) \mapsto v^* \| r^*, \quad \pi(x') \mapsto \perp \quad \big\}. \end{aligned}$$

Lastly, we remove r^* as a random trigger and reach the right-hand side,

$$P = \pi, \qquad y^* = \pi(x'), \qquad Q = \pi^{-1}\{\pi(x') \mapsto \perp\} = \pi^{-1} \setminus \{y^*\}.$$

Adaptive Security. One caveat of the proofs above is that they only work only when v^* (the challenge plaintext x^*) is chosen prior to receiving $\mathsf{pk} \xleftarrow{\$} i\mathcal{O}(\pi)$ and access to the decryption oracle. The scheme can be used as a CCA-secure key encapsulation mechanism (KEM), but we will discuss this point later. Now, we address the issue of adaptive security.

Recall that in the standard definition of CCA security, the adversary sees pk, interacts with the decryption oracle, then chooses the challenge plaintext(s), and further interacts with the decryption oracle before outputting a decision bit. We start by identify two difficulties: *i)* the decryption *oracle* becomes punctured only after choosing the challenge plaintext, but any *program*, once given to the adversary, cannot be retroactively punctured (at least not classically); *ii)* the permutation Γ applied to π must be chosen prior to seeing π in PPRP security, but Γ must also somehow "work on" a plaintext chosen dependent on π.

Our first observation is that the decryption program can be punctured *by encryption randomness*. Namely, we sample $r^* \xleftarrow{\$} \{0,1\}^\lambda$ at the onset of the CCA game, and reject any pre-challenge decryption query ct' such that $\pi^{-1}(\mathsf{ct}')$ ends with r^*. This only leads to a negligible statistical error as the adversary's view is independent of r^* prior to receiving the challenge $\mathsf{ct}^* = y^*$. It removes the apparent capability of decrypting ct^*. Moreover, once v^* is chosen, we reveal another decryption program punctured only at y^*. Together, they solve the first difficulty.

To overcome the second difficulty, we design Γ to randomize *every possible plaintext* when the encryption randomness is r^*. Letting $u^* \xleftarrow{\$} \{0,1\}^{|v^*|}$, $s^* \xleftarrow{\$} \{0,1\}^{\lambda}$, we define

$$\Gamma(v\|r) := \begin{cases} (u^* \oplus v)\|s^*, & \text{if } r = r^*; \\ v\ \|r^*, & \text{if } r = s^*; \\ v\ \|r, & \text{otherwise.} \end{cases}$$

In other words, Γ applies a one-time pad u^* for the challenge and swaps randomness values r^*, s^*. The PPRP of [SZ25] is permutable by such Γ.

Roughly speaking, we prove

$$\big(i\mathcal{O}(P),\, i\mathcal{O}(Q_0),\, y_0,\, i\mathcal{O}(R_0)\big) \approx \big(i\mathcal{O}(P),\, i\mathcal{O}(Q_1),\, y_1,\, i\mathcal{O}(R_1)\big) \tag{3}$$

for v^* adversarially chosen dependent on the first two programs and

$$P = \pi, \qquad Q_0 = \pi^{-1} \setminus \left\{ \pi(v\|r^*) \,\middle|\, v \in \{0,1\}^{|v^*|} \right\},$$
$$y_0 \leftarrow \pi(v^*\|r^*), \qquad y_1 \xleftarrow{\$} \{0,1\}^{|v^*|+\lambda},$$
$$Q_1 = \pi^{-1} \setminus \{y_1\}, \qquad R_\beta(y) = \begin{cases} \bot, & \text{if } y = y_\beta; \\ v, & \text{if } y \neq y_\beta \text{ and } \pi^{-1}(y) = v\|r. \end{cases}$$

Here, we strip the randomness from the output of R due to some technicalities in the proof.

The proof involves two rounds of trigger–permute–untrigger. A general guideline in choosing the functionality on the triggered points is to *(pre-)compose* Γ^{-1} *to* π *on that branch* so that it reverts to π (normal behavior) once π is permuted by Γ. Morphing the hybrids from the left-hand side, for the first round, we first plant trigger s^*:

$$P(v\|r) = \begin{cases} \pi(\Gamma^{-1}(v\|r)) = \pi((u^* \oplus v)\|r^*), & \text{if } r = s^*; \\ \pi(v\|r), & \text{otherwise;} \end{cases}$$
$$Q = \pi^{-1} \setminus \left\{ \pi(v\|r^*),\, \pi(v\|s^*) \,\middle|\, v \in \{0,1\}^{|v^*|} \right\},$$

and R is unchanged. Next, we permute π by Γ; then, we remove random trigger r^*. The overall effect is to change the components to

$$P = \pi, \qquad Q = \pi^{-1} \setminus \left\{ \pi(v\|s^*) \,\middle|\, v \in \{0,1\}^{|v^*|} \right\},$$
$$y^* = \pi((u^* \oplus v^*)\|s^*), \qquad R(y) = \begin{cases} \bot, & \text{if } y = y^*; \\ v \oplus u^*, & \text{else if } \pi^{-1}(y) = v\|s^*; \\ v, & \text{else, } \pi^{-1}(y) = v\|r \text{ for } r \neq s^*. \end{cases}$$

By changing the variable to $w^* \xleftarrow{\$} \{0,1\}^{|v^*|}$ and programming $u^* \leftarrow v^* \oplus w^*$, we write $y^* = \pi(w^*\|s^*)$ without v^*, but the code of R leaks information about v^* (due to u^*).

The current hybrid is similar to the case of selective security — the "plaintext" inside y^* is chosen independently of π. Intuitively, we can delete the leaky branch of R because s^*, although appearing in y^*, should be pseudorandom hence a trigger. Indeed, it can be done by a second round of trigger–permute–untrigger, permuting π by $(w^*\|s' \;\; w^*\|s^*)$ for a random s'. (It also makes Q punctured only at y^*.)

2.2 Free Upgrade to Functional Encryption

The above PKE scheme can be easily upgraded to provide functional decryption capability, without changing the encryption key/algorithm at all. In the customary notation for FE, the PKE scheme is cast to $\mathsf{mpk} \xleftarrow{\$} i\mathcal{O}(\pi)$, $\mathsf{msk} = \pi^{-1}$, $\mathsf{ct}(x) = \pi(x\|r)$, and $\mathsf{Dec}(\mathsf{msk}, \mathsf{ct}) = x$ with $x\|r \leftarrow \pi^{-1}(\mathsf{ct})$. To generate a decryption key for circuit C, obfuscate the decrypt-then-compute program, i.e., $\mathsf{sk}_C \xleftarrow{\$} i\mathcal{O}(\mathsf{ct} \mapsto C(\mathsf{Dec}(\mathsf{msk}, \mathsf{ct})))$.

This simple scheme is *simulation*-secure against chosen-ciphertext attacks (SIM-CCA). More precisely, consider the following (real) interaction. The adversary $\mathcal{A}$ receives mpk and is given access to key creation, key retrieval, and decryption oracles. Here, when a key sk_j for circuit C_j is created, it is *not* automatically revealed to $\mathcal{A}$. Instead, $\mathcal{A}$ is given a handle (index) to it. It can submit the handle to retrieve the key, or submit ct′ with any index to perform decryption using any key, or query for decryptions adaptively and retrieve keys conditionally, thereby modelling a flexible class of attacks. Prior to the first key retrieval query, the adversary must have chosen a challenge plaintext x^*, for which it gets back a ciphertext ct^*.

We construct an efficient simulator such that for any efficient adversary, the real interaction is indistinguishable to the simulation — the simulator is responsible for creating mpk, ct^*, and responding to oracle queries. Upon the challenge, the simulator must output ct^* without being given x^*. All key creation and decryption queries are answered *without* giving $C(x^*)$ to the simulator, regardless of them being pre- or post-challenge. Upon a key retrieval query[4] (to key j for circuit C_j), the value $C_j(x^*)$ is provided to the simulator, which responds with sk_j.

The simulator is straight-forward. It generates mpk normally, keeps π^{-1} to itself, and samples a random string ct^* as the simulated ciphertext. To create key j for circuit C_j, it remembers (j, C_j). To decrypt ct′ using key j, it outputs $C_j(x)$ for $x\|r \leftarrow \pi^{-1}(\mathsf{ct}')$ if $\mathsf{ct}' \neq \mathsf{ct}^*$, or else $\perp$ if $\mathsf{ct}' = \mathsf{ct}^*$ (including in the pre-challenge phase). When a key is to be retrieved for the first time, the simulator gets $z_j^* = C_j(x^*)$, and it outputs $\mathsf{sk}_j \xleftarrow{\$} i\mathcal{O}(\widehat{C}_j)$, where $\widehat{C}_j(\mathsf{ct})$ outputs z_j^* if $\mathsf{ct} = \mathsf{ct}^*$ or else $C_j(\mathsf{Dec}(\pi^{-1}, \mathsf{ct}))$ — in other words, a simulated key hardwires the decryption result for the simulated ciphertext and otherwise operates normally.

The validity of our simulator is shown using Equation (3). The program Q is used to answer decryption queries and R is used to generate retrieved keys.

2.3 Optimized Key Encapsulation Mechanism

Although simple and powerful for extension, the simplistic PKE does not scale well with the plaintext length — the major overhead would be obfuscating large circuits with long inputs. In practice, public-key encryption is almost always done using hybrid encryption. Any PKE can be used as a KEM, albeit at the cost of ciphertext blow-up on top of *true* entropy in the encapsulated key, i.e., $\mathsf{Enc}(\mathsf{pk}, k; r)$ is necessarily longer than k, which has at most $|k|$ bits of true entropy.

We construct a simple *size-optimized* KEM by obfuscating *two* PPRP instances. Let π, σ be two PPRP over domain $\{0,1\}^\lambda$. We set $\mathsf{pk} \xleftarrow{\$} (i\mathcal{O}(\pi), i\mathcal{O}(\sigma))$ and $\mathsf{sk} = (\pi^{-1}, \sigma)$.

[4] It is stressed that functional keys can be *retrieved* only after the challenge plaintext x^* is chosen. Indeed, if an unbounded number of key retrievals are allowed prior to choosing x^*, simulation-secure FE for general circuits is impossible in the standard model [BSW11,AGVW13]. On the other hand, they can be (created and) *used for decryption queries* in both pre- and post-challenge phases.

A ciphertext is $\mathsf{ct} = \pi(r)$ and it encapsulates key $k = \sigma(r) = \sigma(\pi^{-1}(\mathsf{ct}))$. Note that the ciphertext length is $|\mathsf{ct}| = \lambda$ and k, marginally, has λ bits of entropy, i.e., there is no blow-up at all.

The scheme is CCA-secure, which boils down to

$$\begin{aligned} &\big(i\mathcal{O}(\pi),\, i\mathcal{O}(\sigma),\, \pi(r),\, \sigma(r),\;\; i\mathcal{O}(\pi^{-1} \setminus \{\pi(r)\})\big) \\ \approx\; &\big(i\mathcal{O}(\pi),\, i\mathcal{O}(\sigma),\, \pi(s),\, \sigma(s'),\, i\mathcal{O}(\pi^{-1} \setminus \{\pi(s)\})\big), \end{aligned} \tag{4}$$

where r, s, s' are independent λ-bit random strings. Here, the last program (together with $i\mathcal{O}(\sigma)$) is used to answer decryption queries. The proof is a simple extension of PKE selective security: we plant s (resp. s') into π and π^{-1} (resp. σ), then permute π by $(r \;\; s)$ and σ by $(r \;\; s')$.

On top of having full-entropy keys, assuming subexponential $i\mathcal{O}$ and PPRP, the *concrete* computational security of this KEM is *only* limited by the security level of $|\mathsf{ct}|$-bit triggers. Observe that dialing up the security parameters of $i\mathcal{O}$ and PPRP only affects the lengths of pk, sk, not those of ct, k.

Cohardcore Functions. The KEM result can be phrased as "$i\mathcal{O}(\sigma)$ is a hardcore function for $i\mathcal{O}(\pi)$, even in the presence of punctured π^{-1}". A consequence of the plaintext-selective PKE result is that any subset of preimage bits (corresponding to the plaintext) are hardcore as long as the complement subset is large enough. One could ask what other functions h are hardcore for $i\mathcal{O}(\pi)$. We note, however, that a better notion to consider is *cohardcoreness*.

Recall that h is *hardcore* for f if $h(x)$ is pseudorandom given $f(x)$, with the most interesting case being $f(x)$ determining $h(x)$. We say that h is *cohardcore* for f if $f(x)$ is pseudorandom given $h(x)$, with the interesting case *still* being $f(x)$ determining $h(x)$. In this terminology, plaintexts are cohardcore for ciphertexts in PKE. The reason to consider cohardcoreness is that in general, nothing specific can be said about the image distribution of an arbitrary function h, yet $i\mathcal{O}(\pi)$ is an obfuscated PPRP, whose image we can expect to be pseudorandom. This dual notion can often be converted to the primal — if h has marginally pseudorandom image distribution and it is cohardcore for f, then it is also hardcore for f.

We prove that a large class of functions h are cohardcore for obfuscated PPRP $i\mathcal{O}(\pi)$, even for non-uniform input distribution (subject to conditions) and even in the presence of punctured inverse of π. The condition for h and input x^* is that a public polynomial-size circuit h' exists so that $(h(x^*), h'(x^*))$ has unique preimage x^* under the tuple map $x \mapsto (h(x), h'(x))$ and $h'(x^*)$ is long enough and pseudorandom (given the other information). The function h' will serve the "untrigger" step. This result is general enough to recover KEM and plaintext-selective PKE, as well as a lemma that will be used for unary homomorphic encryption.

2.4 Unary Homomorphic Encryption

Taking another route, we show how to upgrade the same PKE scheme to enable homomorphic evaluation over encrypted data. This is probably the most surprising part of this work. More specifically, given any circuit $g : \{0,1\}^n \to \{0,1\}^n$, we augment the public key with an obfuscation of $\mathsf{Ev}_g(\pi(x\|r)) = \pi(g(x)\|\sigma'(r))$ while preserving semantic security. For adversarially chosen $g : \{0,1\}^n \to \{0,1\}^n$ and $x^* \in \{0,1\}^n$, let

$r^* \xleftarrow{\$} \{0,1\}^\lambda$, we show

$$\big(i\mathcal{O}(\pi),\ i\mathcal{O}(\mathsf{Ev}_g),\ \pi(x^*\|r^*)\big) \approx \big(i\mathcal{O}(\pi),\ i\mathcal{O}(\mathsf{Ev}_g),\ \text{random}\big). \tag{5}$$

We now describe how Ev_g (in particular, σ') works and how security is proven.

For technicalities in the proof, we conceptually split the encryption randomness into two parts, a counter ℓ and *the* randomness r. For convenience, let each be λ bits. We define $\sigma'(\ell\|r) := (\ell+1)\|\sigma(r)$, where "$+1$" is modulo 2^λ and σ is another instance of PPRP for rerandomization.

Spelled out more completely, the public key is $(i\mathcal{O}(\pi), i\mathcal{O}(\mathsf{Ev}_g))$ and a ciphertext is $\mathsf{ct}(x) = \pi(x\|\ell\|r)$. The program $\mathsf{Ev}_g(\mathsf{ct})$ computes $x\|\ell\|r \leftarrow \pi^{-1}(\mathsf{ct})$ and then outputs $\pi(g(x)\|(\ell+1)\|\sigma(r))$. To evaluate $g^{(t)}$ on ct, iterate $i\mathcal{O}(\mathsf{Ev}_g)$ for t times on ct. It is stressed that the encryption algorithm remains unchanged — in a fresh ciphertext, both ℓ and r are random.

Proving security (Equation (5)) in the presence of $i\mathcal{O}(\mathsf{Ev}_g)$ is more challenging than the basic case (even with CCA, Equation (2)). Firstly, Ev_g is internally capable of decrypting every possible ciphertext, so there is already some flavor of CCA going on. Secondly, the triple $(i\mathcal{O}(\pi), i\mathcal{O}(\mathsf{Ev}_g), \mathsf{ct}^*)$, where $\mathsf{ct}^* = \pi(x^*\|\ell^*\|r^*)$, is in essence the compressed form of a very long chain $\mathsf{ct}^* = \mathsf{ct}^{(0)} \to \mathsf{ct}^{(1)} \to \cdots$ where $\mathsf{ct}^{(t)} = \pi(g^{(t)}(x^*)\|(\ell^*+t)\|\sigma^{(t)}(r^*))$ and its length is not bounded by any polynomial. We are proving security for a *super-polynomial* number of ciphertexts with *correlated* encryption randomness.

Observe that every evaluation chain eventually forms a ρ-shaped cycle. We plant a random trigger $\widehat{\ell^*}$ to make $\mathsf{Ev}_g(\mathsf{ct})$ output $\perp$ when $x\|\ell\|r \leftarrow \pi^{-1}(\mathsf{ct})$ has $\ell = \ell^* + \widehat{\ell^*}$. This "cuts" the chain from ct^* down to a manageable length:

$$\begin{aligned}
&\text{(before)}\quad \mathsf{ct}^* = \mathsf{ct}^{(0)} \to \mathsf{ct}^{(1)} \to \cdots \to \mathsf{ct}^{(\widehat{\ell^*})} \xrightarrow{\mathsf{Ev}_g} \mathsf{ct}^{(\widehat{\ell^*}+1)} \to \cdots && \text{(eventually cycles);}\\
&\text{(after)}\quad \mathsf{ct}^* = \mathsf{ct}^{(0)} \to \mathsf{ct}^{(1)} \to \cdots \to \mathsf{ct}^{(\widehat{\ell^*})} \xrightarrow{\mathsf{Ev}_g} \perp && \text{(never cycles).}
\end{aligned}$$

We then go through $\widehat{\ell^*}$ hybrids to remove x^* from this chain, by replacing ct's on a longer and longer prefix by pseudorandom values. We use yet another PPRP φ to derive the fake ciphertexts. A typical hybrid looks like

$$\mathsf{ct}^* = \varphi(0) \to \varphi(1) \to \cdots \to \boxed{\varphi(\mathsf{t})} \xrightarrow{\mathsf{Ev}_g} \boxed{\mathsf{ct}^{(\mathsf{t}+1)}} \to \cdots \to \mathsf{ct}^{(\widehat{\ell^*})} \to \perp.$$

Here, Ev_g uses φ^{-1}, φ to recognize/generate the fake ciphertexts. Advancing t appears a bit more involved, as there is a deep correlation among the r's inside the suffix ct's via σ. Let v^* be adversarially chosen. We rely on

$$\big(i\mathcal{O}(\pi),\ i\mathcal{O}(\pi^{-1}\setminus\{y_0\}),\ i\mathcal{O}(\sigma),\ y_0,\ s_0\big) \approx \big(i\mathcal{O}(\pi),\ \underbrace{i\mathcal{O}(\pi^{-1}\setminus\{y_1\}),\ i\mathcal{O}(\sigma)}_{\text{for } \mathsf{Ev}_g},\ \underbrace{y_1,\ s_1}_{\mathsf{ct}^{(\mathsf{t}+1)},\,\mathsf{ct}^{(\mathsf{t}+2)}}\big), \tag{6}$$

where $y_0 = \pi(v^*\|r^*)$, $s_0 = \sigma(r^*)$, and y_1, s_1 are independent and random. This converts $\mathsf{ct}^{(\mathsf{t}+1)}$ to random without affecting $\mathsf{ct}^{(\mathsf{t}+2)}$. The now-random $\mathsf{ct}^{(\mathsf{t}+1)}$ can be switched to pseudorandom $\varphi(\mathsf{t}+1)$, hence "collected" into the replaced prefix, via

$$\big(i\mathcal{O}(\varphi\setminus\{v^*\}),\ i\mathcal{O}(\varphi^{-1}\setminus\{y_0\}),\ y_0\big) \approx \big(i\mathcal{O}(\varphi\setminus\{v^*\}),\ i\mathcal{O}(\varphi^{-1}\setminus\{y_1\}),\ y_1\big), \tag{7}$$

where $y_0 = \varphi(v^*)$ and y_1 is random. The proof completes once the entire chain is replaced.

We note that Equation (6) is a generalization of Equation (4) — the latter is the former with v^* being the empty string. Both can be derived from the cohardcoreness result. Moreover, Equation (7) says that obfuscations of *permutable* PRP with large domain are also *puncturable*, under a suitable definition of puncturable PRP.[5]

Challenges to Realizing Richer Homomorphisms. It is tempting to consider extending the current scheme by adding homomorphic evaluation gadgets that combine two or more ciphertexts into a single ciphertext, where the corresponding plaintexts are meaningfully correlated, as in traditional binary (or multi-ary) FHE schemes. A less ambitious, yet still interesting, extension is to keep the current unary structure, but allow multiple homomorphic evaluation programs that work with a single public key.

However, while we are not aware of concrete attacks against those extensions, we are unable to assert their security. Let us provide some perspective for this inability, by considering a unary HE scheme with *two* homomorphic evaluation programs Ev_{g_0} and Ev_{g_1} for functions $g_0, g_1 : \{0,1\}^n \to \{0,1\}^n$. Such a scheme enables homomorphic evaluation of $g_{b_t} \circ \cdots \circ g_{b_2} \circ g_{b_1}$ for any $b_1, \ldots, b_t \in \{0,1\}$ and for t not *a priori* bounded by any polynomial.[6] This is a significantly more complex requirement than Equation (5). Instead of having a single chain of t ciphertexts originating from ct^*, we now have a depth-t binary tree rooted at ct^*, of 2^t ciphertexts with correlated encryption randomness. Reductions cannot afford to incur loss per each one of them (up to $2^{\lambda^{\omega(1)}}$). Prior works dealt with a similar situation using *lossy* encryption together with leveled homomorphism [CLTV15]. Such techniques are incompatible with our unleveled, permutation-based setting.

We further substantiate the gap between single-function and two-function unary HE by demonstrating that proving security for two functions necessarily either relies on primitives other than PPRP and $i\mathcal{O}$ or involves a breakthrough in non-black-box techniques. We prove that unary HE for two-function iterations implies, in black box, primitives known to be unobtainable from black-box usage of $i\mathcal{O}$ and one-way functions. Consider

$$g_b(i^*, i, v) = \begin{cases} (i^*, i+1, b), & \text{if } i+1 = i^*; \\ (i^*, i+1, v), & \text{if } i+1 \neq i^*; \end{cases} \qquad \text{for } b \in \{0,1\},$$

which implements "read the bit at index i^*" in a streaming fashion, i.e.,

$$(g_{b_t} \circ \cdots \circ g_{b_2} \circ g_{b_1})(i^*, 0, 0) = (i^*, t, b_{i^*}).$$

Such a unary HE immediately implies the following.

- *One-Round Private Information Retrieval.* The server stores the database $x_1 \cdots x_t$ and $(\mathsf{pk}, \mathsf{Ev}_{g_0}, \mathsf{Ev}_{g_1})$ from the set-up phase. To query at index i^*, the client sends $\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, (i^*, 0, 0))$. The server responds with the homomorphically evaluated $\mathsf{Ev}_{g_{x_t}}(\cdots \mathsf{Ev}_{g_{x_2}}(\mathsf{Ev}_{g_{x_1}}(\mathsf{ct}))\cdots)$, which the client can decrypt to x_{i^*}.
- *Somewhere Extractable Hash.* Here, $\mathsf{hk} = (\mathsf{pk}, \mathsf{Ev}_{g_0}, \mathsf{Ev}_{g_1}, \mathsf{ct})$ is the hash key and $H_{\mathsf{hk}}(x_1 \cdots x_t) = \mathsf{Ev}_{g_{x_t}}(\cdots \mathsf{Ev}_{g_{x_2}}(\mathsf{Ev}_{g_{x_1}}(\mathsf{ct}))\cdots)$. If $\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, (i^*, 0, 0))$, then x_{i^*} is extractable from the hash value using sk.

[5] The work of [BKW17] proposed a different definition of puncturable PRP and showed that it cannot be achieved. See Remark 5 for details.

[6] When t is bounded by a fixed polynomial, or more precisely, when the ciphertext length can grow polynomially in t (and λ), this is just $i\mathcal{O}$ for compute-and-encrypt.

Neither application can be obtained from just $i\mathcal{O}$ and OWF in black box [AS15,IKO05], so any extension of our scheme to support two-function iterations requires additional primitives or breakthrough non-black-box techniques.

2.5 Instantiation with Reversible Circuits

We substantiate an alternative route to PPRP and strong PKE/KEM, based on specific properties of random reversible circuits. Let's start with some background.

Our starting point is the fact that the family of permutations computed by random n-wire m-gate[7] reversible circuits are ε-close to t-wise independent ones for some $m = \widetilde{O}(nt \log(1/\varepsilon))$ [Gow96,GHP25]. Furthermore, it has been conjectured that they are in fact a family of strong PRP. Extending from this conjecture, the work of [CCMR24] showed that if one could, given a random reversible circuit C of certain polynomial size, come up with with a functionally equivalent one that is "sufficiently rerandomized" relative to C, then one could also perform general-purpose $i\mathcal{O}$ for all (reversible or not) circuits. They also propose a host of mechanisms for reranodmizing random reversible circuits by way of repeatedly performing local, functionality-preserving perturbations. Two points from [CCMR24] are of interest for the present work.

First, their work constructs an $i\mathcal{O}$ that preserves the format of reversible circuits (termed rc$i\mathcal{O}$ by us), i.e., for an n-wire reversible circuit C, its obfuscation rc$i\mathcal{O}(C)$ is a functionally equivalent n-wire reversible circuit, and in particular, allows evaluation of C^{-1}. To get $i\mathcal{O}$ for normal circuits, a circuit must be transformed into a reversible one with many ancillae, a somewhat costly process that we would like to avoid.

Second, along the way to rc$i\mathcal{O}$, the work of [CCMR24] formulates a computational hardness assumption, which turns out to be very relevant for our study. Their *split-circuit pseudorandomness* (SCP) assumption pertains to the nature of reversible circuits. In a simplified form, it states that there exists $m^{\#} \in \mathrm{poly}(n, \lambda)$ such that for any large enough n and two n-wire reversible circuits C_1, C_2, the following two distributions are computationally indistinguishable. The first distribution is uniformly random over all $m^* = (|C_1| + 2m^{\#} + |C_2|)m^{\#}$-gate circuits functionally equivalent to $C_1 \| C_2$. (This distribution can be thought of as an "ideal" obfuscation of $C_1 \| C_2$ to m^*-gate circuits.) The second distribution first samples a random n-wire $m^{\#}$-gate circuit R, next a uniformly random $(|C_1| + m^{\#})m^{\#}$-gate circuit $\widehat{C}_1$ functionally equivalent to $C_1 \| R$, then a uniformly random $(m^{\#} + |C_2|)m^{\#}$-gate $\widehat{C}_2$ equivalent to $R^{-1} \| C_2$, and lastly outputs $\widehat{C}_1 \| \widehat{C}_2$.

PPRP from Split-Circuit Pseudorandomness and rc$i\mathcal{O}$. Assuming SCP and rc$i\mathcal{O}$, we show that a random n-wire m-gate reversible circuit R is a PRP permutable by polynomial-size n-wire reversible circuits, for which a Γ-permuted key is simply rc$i\mathcal{O}(\Gamma \| R)$ (evaluatable both ways). Essentially, SCP provides that any fixed circuit ($C_1 = \Gamma$) can be smudged into random m-gate ones (R) given sufficient expansion ($m^{\#}$), but its distributions are not efficiently sampleable. An rc$i\mathcal{O}$ canonicalizes them into efficient ones.

TDP/PKE/KEM from Reversible-Circuit PPRP and $i\mathcal{O}$. Due to the format-preserving nature, rc$i\mathcal{O}(R)$ cannot possibly be the pk of a secure TDP (or PKE or KEM) because

[7] Each gate is of the form $x_k \leftarrow x_k \oplus f(x_i, x_j)$ for distinct wires $i, j, k \in [n]$ and some $f : \{0,1\}^2 \to \{0,1\}$.

it enables efficient inversion of R. The simplest fix is to obfuscate its Toffoli version, $\mathsf{TOF}[R](x,y) = (x, y \oplus R(x))$, a $2n$-wire circuit consisting of applying R to the first n wires, applying CNOT's to XOR the first n wires to the last n wires, and lastly applying R^{-1} to restore the first n wires — $\mathsf{TOF}[R]$ is its own inverse and does not appear to help with inverting R. Note that there is no additional ancillae in $\mathsf{TOF}[R]$. It is *a priori* unclear whether the proof of one-wayness or CCA security can go through in this setting, due to the complex surgeries applied to the circuits in the hybrids.

We prove that the corresponding KEM is still CCA-secure (and for PKE, plaintext-selectively CCA-secure). Along the way, we develop reversible-circuit triggers and work around the issue of transposition being odd (reversible circuits with at least four wires only compute even permutations).

3 Preliminaries

The security parameter is λ, omitted for brevity except in definitions. A function $f : \mathbb{N} \to \mathbb{R}$ is negligible if $f = \mathrm{O}(\lambda^{-c})$ for all $c \in \mathbb{N}$. We denote by $\mathrm{negl}(\cdot)$ an unspecified negligible function and by $\mathrm{poly}(\cdot)$ a polynomially bounded one. Efficient means probabilistic polynomial-time in the total input length. Adversaries might also be given advice dependent on λ (length bounded by time complexity).

Two experiments Exp_0, Exp_1 with single-bit output are (computationally) indistinguishable (resp. statistically indistinguishable; identical), denoted by $\approx$ (resp. $\approx_s$; $\equiv$), if $(\Pr[\mathsf{Exp}_0(1^\lambda, \mathcal{A}) \to 1] - \Pr[\mathsf{Exp}_1(1^\lambda, \mathcal{A}) \to 1])$ is negligible for all efficient $\mathcal{A}$ (resp. all $\mathcal{A}$; the zero function for all $\mathcal{A}$). They are subexponentially indistinguishable if for some $c > 0$, for all $\mathcal{A}$ running in time $\mathrm{O}(2^{\lambda^c})$, that difference is $\mathrm{O}(2^{-\lambda^c})$. (See Remark 11 on "subexponential".)

The concatenation of strings x, y is denoted by $x \| y$. Intervals are intersected with $\mathbb{Z}$, e.g., $(a, b]$ is $\{a+1, a+2, \ldots, b-1, b\}$ (and $\varnothing$ if $a \geq b$) when $a, b \in \mathbb{Z}$. We also write $[a]$ for $[1, a]$. Given a program P, we write $P \setminus S$ for P modified (in some standard manner) to output $\perp$ whenever the input is in S. Formally,

$$(P \setminus S)(x) := \begin{cases} \perp, & \text{if } x \in S; \\ P(x), & \text{otherwise.} \end{cases}$$

Symbols. Table 1 explains select single-letter symbols used in this paper.

Definition 1 (indistinguishability obfuscation [BGI+12]). A *circuit obfuscator* is an efficient algorithm $i\mathcal{O}(1^\lambda, C)$ transforming a circuit $C : \{0,1\}^n \to \{0,1\}^m$ into one with identical functionality, i.e., for all such C and input $x \in \{0,1\}^n$, it holds that $\Pr[i\mathcal{O}(1^\lambda, C)(x) = C(x)] = 1$. The obfuscator satisfies *indistinguishability-based security* if $\mathsf{Exp}^0_{\mathrm{io}} \approx \mathsf{Exp}^1_{\mathrm{io}}$, where $\mathsf{Exp}^\beta_{\mathrm{io}}(1^\lambda, \mathcal{A})$ proceeds as follows.

- **Challenge.** Launch $\mathcal{A}(1^\lambda)$. Receive two circuits $C_0, C_1 : \{0,1\}^n \to \{0,1\}^m$ from it. Run $\widetilde{C} \xleftarrow{\$} i\mathcal{O}(1^\lambda, C_\beta)$ and send $\widetilde{C}$ to $\mathcal{A}$.
- **Guess.** $\mathcal{A}$ outputs a bit β'. The output of the experiment is β' if C_0, C_1 are of the same size and have identical functionality; otherwise, the output is set to 0.

Definition 2 (permutable PRP [SZ25]). Let $G = \{G_{\lambda,N}\}_{\lambda,N \in \mathbb{N}}$ be a family of non-empty sets of permutations such that $\Gamma : \{0,1\}^N \to \{0,1\}^N$ for all $\Gamma \in G_{\lambda,N}$. An *input-permutable PRP* (IP-PRP) for G consists of four efficient algorithms.

- $\mathsf{Gen}(1^\lambda, 1^N)$, given the domain length N, outputs a (non-permuted) key π.
- $\mathsf{Permute}(1^\lambda, \pi, \Gamma, b)$ takes as input a non-permuted key π, some $\Gamma \in G_{\lambda,N}$, and a bit $b \in \{0,1\}$. It outputs a permuted key $\pi_{\Gamma,b}$.
- $\mathsf{Eval}(1^\lambda, \pi, x)$ takes as input a (permuted or not) key π and an input $x \in \{0,1\}^N$. It is deterministic and outputs some $y \in \{0,1\}^N$.
- $\mathsf{Eval}^{-1}(1^\lambda, \pi, y)$ takes as input π and $y \in \{0,1\}^N$. It outputs some $x \in \{0,1\}^N$.

The scheme must be *correct*, i.e., for all $\lambda, N \in \mathbb{N}$, $x \in \{0,1\}^N$, $\Gamma \in G_{\lambda,N}$, $b \in \{0,1\}$, it holds that

$$\Pr\left[\begin{array}{l} \pi \xleftarrow{\$} \mathsf{Gen}(1^\lambda, 1^N) \\ \pi_{\Gamma,b} \xleftarrow{\$} \mathsf{Permute}(1^\lambda, \pi, \Gamma, b) \end{array} : \begin{array}{r} \mathsf{Eval}^{-1}(1^\lambda,\ \pi\ , \mathsf{Eval}(1^\lambda,\ \pi\ , x)) = x \\ \text{and } \mathsf{Eval}^{-1}(1^\lambda, \pi_{\Gamma,b}, \mathsf{Eval}(1^\lambda, \pi_{\Gamma,b}, x)) = x \\ \text{and } \mathsf{Eval}(1^\lambda, \pi_{\Gamma,b}, x) = \mathsf{Eval}(1^\lambda, \pi, \Gamma^b(x)) \end{array}\right] = 1,$$

where Γ^b is the bth iteration of Γ.

The scheme is *secure* if $\mathsf{Exp}^0_{\text{pprp}} \approx \mathsf{Exp}^1_{\text{pprp}}$, where $\mathsf{Exp}^\beta_{\text{pprp}}(1^\lambda, \mathcal{A})$ proceeds as follows.

- **Challenge.** Launch $\mathcal{A}(1^\lambda)$. Receive from it 1^N and $\Gamma \in G_{\lambda,N}$. Run

  $$\pi \xleftarrow{\$} \mathsf{Gen}(1^\lambda, 1^N), \qquad \pi_{\Gamma,\beta} \xleftarrow{\$} \mathsf{Permute}(1^\lambda, \pi, \Gamma, \beta),$$

  and send $\pi_{\Gamma,\beta}$ to $\mathcal{A}$.
- **Guess.** $\mathcal{A}$ outputs a bit, which is the output of the experiment.

context	symbol	meaning
generic	λ, β	security parameter, challenge bit
	C, c	circuit, constant for subexponential security
	$\not{\Delta}$, Δ	distributions of all-zero / point-function circuits
	$\mathcal{T}$	trigger (an all-zero or point-function circuit)
	x, v	plaintext/input, adversarially chosen value
	r, s, u, w	randomness ($\times 2$), one-time pad ($\times 2$)
	$\widetilde{C}$	program (obfuscated from C)
	P, Q, R	programs being obfuscated
PPRP	π, σ, φ	instances of PPRP
	Γ, $(x\ \ x')$	permutation, transposition
FE	q, $\widehat{C}$	count of queries, circuit (modified from C)
	j, J	index / count of functional keys
HE	D, H, h	domain, function set, function
	g	generator (step function)
	t, ℓ	number of steps, counter (level)
reversible circuits	n, m	number of wires / gates
	$\mathcal{C}_{n,m}$, $\mathcal{C}_{n,\star}$	set of reversible circuits
	$\mathcal{E}$	distribution of random equivalent circuit
	$\widehat{C}$	circuit equivalent to C from $\mathcal{E}$

Table 1. Select single-letter symbols.

For simplicity, when the IP-PRP scheme in use is clear from context, we write π, π^{-1} for $\mathsf{Eval}(\pi, \cdot)$ and $\mathsf{Eval}^{-1}(\pi, \cdot)$, respectively, i.e., treat a key as the permutation determined by it. We may write π as π^{-1} in the context of bit-strings (i.e., the string π^{-1} is exactly the string π), notably in secret keys, to emphasize its usage for inverse evaluation in that context. We assume, without loss of generality, that Eval^{-1} (π^{-1}) is guarded against incorrect input lengths[8] by outputting $\perp$ whenever $y \notin \{0,1\}^N$.

The work of [SZ25] constructed IP-PRP for "decomposable" permutations. We rely on the following kinds of permutations, which are decomposable as defined in [SZ25].

Lemma 1 (PPRP [SZ25]). *Based on one-way functions and iO (Definition 1), both with subexponential security, one can construct a subexponentially secure IP-PRP (Definition 2) for*

$$G_N = \left\{ (x \;\; x') \,\middle|\, x, x' \in \{0,1\}^N \right\} \cup \left\{ \mathsf{URS}[u', r', s'] \,\middle|\, N' \in [0, N],\ u' \in \{0,1\}^{N'},\ r', s' \in \{0,1\}^{N-N'} \right\},$$

where[9]

$$\mathsf{URS}[u', r', s'](v \| r) := \begin{cases} (u' \oplus v) \| s', & \text{if } r = r'; \\ v \;\| r', & \text{if } r = s' \text{ and } r \neq r'; \\ v \;\| r\,, & \text{otherwise;} \end{cases}$$

for $v \in \{0,1\}^{N-N'}$ and $r \in \{0,1\}^{N'}$.

Random Triggers. The technique of random triggers is heavily used. We formalize it as the following lemma.

Lemma 2 (random triggers; ¶). *Based on one-way functions and iO (Definition 1), one can construct a* trigger scheme*, consisting of two efficient algorithms as follows.*

- $\not\Delta(1^\lambda, 1^N)$ *takes the input length N as input and outputs a circuit computing $\mathcal{O}_N$, the zero function over $\{0,1\}^N$, defined as $\mathcal{O}_N(x) := 0$ for all $x \in \{0,1\}^N$.*
- $\Delta(1^\lambda, r)$ *takes the trigger r as input and outputs a circuit computing δ_r, the point function at r over $\{0,1\}^{|r|}$, defined as*

$$\delta_r(x) := \begin{cases} 1, & \text{if } x = r; \\ 0, & \text{otherwise;} \end{cases} \qquad \text{for } x \in \{0,1\}^{|r|}.$$

The scheme is correct*, i.e., for all $\lambda \in \mathbb{N}$, $r \in \{0,1\}^\star$, $x \in \{0,1\}^{|r|}$, it holds that*

$$\Pr\left[\not\Delta(1^\lambda, 1^{|r|})(x) = \mathcal{O}_{|r|}(x)\right] = \Pr\left[\Delta(1^\lambda, r)(x) = \delta_r(x)\right] = 1.$$

The scheme is also secure*, i.e., $\mathsf{Exp}^0_{\text{trigger}} \approx \mathsf{Exp}^1_{\text{trigger}}$, where $\mathsf{Exp}^\beta_{\text{trigger}}(1^\lambda, \mathcal{A})$ proceeds as follows.*

[8] This guard is important for CCA security as otherwise, querying the decryption oracle on ciphertexts (inputs) of incorrect lengths could lead to arbitrary behavior unconstrained by correctness/security.

[9] The name URS just indicates its parameter names and has no special meaning. The transpositions are included for explicitness, as $(x \;\; x')$ is simply $\mathsf{URS}[, x, x']$ (u' being empty). We abuse the notation, consistently with URS, so that $(x \;\; x)$ is the identity.

- **Challenge.** Launch $\mathcal{A}(1^\lambda)$ and receive 1^N from it. Run

$$r \xleftarrow{\$} \{0,1\}^N, \qquad \widetilde{C}_0 \xleftarrow{\$} \mathbb{A}(1^\lambda, 1^N), \qquad \widetilde{C}_1 \xleftarrow{\$} \Delta(1^\lambda, r),$$

  and send $\widetilde{C}_\beta$ to $\mathcal{A}$.

- **Guess.** $\mathcal{A}$ outputs a bit β'. The output of the experiment is β' if $N \geq \lambda$; otherwise, the output is set to 0.

Moreover, it is subexponentially secure (see Remark 11) if so are the ingredients.

For completeness, we prove Lemma 2 in Section A. To simplify notation, $\mathbb{A}(1^N)$ may be written as $\mathbb{A}_N$, or just $\mathbb{A}$ when N is clear from context.

4 Functional Encryption with SIM-CCA Security (+ PKE)

In this section, we present a simple construction of functional encryption scheme for circuits with semi-adaptive simulation security against adaptive chosen-ciphertext attacks. It turns out that this is the first FE scheme for general circuits with (any kind of) security against adaptive access to the decryption oracle (see also Footnote 2). By ignoring certain algorithms of the FE scheme, we obtain a CCA-secure PKE scheme as a corollary. (The construction/proof would not be simpler even if we just wanted CCA-secure PKE.)

The scheme has the same form as the trapdoor permutation of [SZ25], yet its proof uses permutability in a more involved way to handle adaptive security. There are some interesting properties about this scheme. Let the plaintext length be n, then a ciphertext is $(n + \lambda)$ bits long and, with the plaintext fixed, have λ bits of entropy (due to encryption randomness). This implies that there is no room for any consistency check information. Indeed, every $(n + \lambda)$-bit string is a valid ciphertext and decryption performs no checks. Decrypting a mauled ciphertext simply yields garbage.

We first recall the syntax of FE for circuits, then define its SIM-CCA security (semi-adaptive simulation, adaptive decryption). After that, we present the construction and its security proof based on IP-PRP and $i\mathcal{O}$. We refer the readers to Section 8 for an instantiation of our CCA-secure PKE by obfuscating reversible circuits.

Definition 3 (FE for circuits). A *functional encryption (FE) scheme for circuits* consists of four efficient algorithms.

- $\mathsf{Setup}(1^\lambda, 1^n)$ takes the message length n as input. It outputs a pair $(\mathsf{mpk}, \mathsf{msk})$ of master public/secret keys.

- $\mathsf{KeyGen}(1^\lambda, \mathsf{msk}, C)$ takes as input msk and a circuit $C : \{0,1\}^n \to \{0,1\}^\star$. It outputs a functional key sk_C.

- $\mathsf{Enc}(1^\lambda, \mathsf{mpk}, x)$ takes as input mpk and a message $x \in \{0,1\}^n$. It outputs a ciphertext ct.

- $\mathsf{Dec}(1^\lambda, \mathsf{msk}, \mathsf{ct})$ or $\mathsf{Dec}(1^\lambda, C, \mathsf{sk}_C, \mathsf{ct})$ takes as input either msk (first form) or C, sk_C (second form), and ct (both forms). It is supposed to recover x (first form) or $C(x)$ (second form).

The scheme must be *correct*, i.e., for all $\lambda \in \mathbb{N}$, $x \in \{0,1\}^\star$, circuit $C : \{0,1\}^{|x|} \to \{0,1\}^\star$, it holds that

$$\Pr\left[\begin{array}{rc:c} (\mathsf{mpk},\mathsf{msk}) & \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^{|x|}) & \mathsf{Dec}(1^\lambda,\mathsf{msk},\mathsf{ct}) = x \\ \mathsf{sk}_C & \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda,\mathsf{msk},C) & \text{and} \\ \mathsf{ct} & \xleftarrow{\$} \mathsf{Enc}(1^\lambda,\mathsf{mpk},x) & \mathsf{Dec}(1^\lambda,C,\mathsf{sk}_C,\mathsf{ct}) = C(x) \end{array}\right] = 1.$$

Definition 4 (SIM-CCA for FE). An FE scheme for circuits (Definition 3) can be equipped with a stateful algorithm Sim whose total running time in $\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{fe}}$ is $\mathrm{poly}(\lambda + n + J_+ + \sum_{j'} |C_{j'}|)$, where J_+ is the total number of queries (of all types). Such a scheme satisfies *SIM-CCA security* if $\mathsf{Exp}^{\mathsf{real}}_{\mathsf{fe}} \approx \mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{fe}}$, where the experiments proceed as follows.

- **Setup.** Launch $\mathcal{A}(1^\lambda)$ and receive 1^n from it. In $\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{fe}}$, launch $\mathsf{Sim}(1^\lambda)$ and maintain its state across the invocations. Run

  $$\begin{array}{rc} \text{(real)} & (\mathsf{mpk},\mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^n), \\ \text{(ideal)} & \mathsf{mpk} \xleftarrow{\$} \mathsf{Sim}(\text{“setup”}, 1^n), \end{array}$$

  and send mpk to $\mathcal{A}$.

- **Query I.** Repeat the following for arbitrarily many rounds determined by $\mathcal{A}$. In each round, $\mathcal{A}$ has two choices (key creation, decryption).

  - $\mathcal{A}$ can submit a circuit $C_J : \{0,1\}^n \to \{0,1\}^\star$ to create a key. Upon such a query, run

    $$\begin{array}{rc} \text{(real)} & \mathsf{sk}_J \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda,\mathsf{msk},C_J), \\ \text{(ideal)} & \mathsf{Sim}(\text{“create”}, C_J), \end{array}$$

    but send nothing to $\mathcal{A}$. Here, J is the total number of key creation queries so far (including this one).
  - $\mathcal{A}$ can submit a ciphertext ct′ and an index $j' \in [0, J]$ for decryption, where J is the total number of key creation queries so far. Upon such a query, run

    $$z \xleftarrow{\$} \begin{cases} \mathsf{Dec}(1^\lambda,\mathsf{msk},\mathsf{ct}'), & \text{(real) if } j' = 0; \\ \mathsf{Dec}(1^\lambda, C_{j'}, \mathsf{sk}_{j'}, \mathsf{ct}'), & \text{(real) if } j' \in [J]; \\ \mathsf{Sim}(\text{“decrypt”}, j', \mathsf{ct}'), & \text{(ideal) always;} \end{cases}$$

    and send z to $\mathcal{A}$.

- **Challenge.** $\mathcal{A}$ chooses $x^* \in \{0,1\}^n$. Upon the challenge, run

  $$\begin{array}{rc} \text{(real)} & \mathsf{ct}^* \xleftarrow{\$} \mathsf{Enc}(1^\lambda,\mathsf{mpk},x^*), \\ \text{(ideal)} & \mathsf{ct}^* \xleftarrow{\$} \mathsf{Sim}(\text{“encrypt”}), \end{array}$$

  and send ct^* to $\mathcal{A}$.

- **Query II.** Similar to Query I, except that $\mathcal{A}$ now has *three* choices (key creation, decryption, key retrieval).

- Key creation remains identical to Query I.
- For decryption: if $\mathsf{ct}' = \mathsf{ct}^*$, the response is set to $\perp$ in both $\mathsf{Exp}^{\mathrm{real}}_{\mathrm{fe}}$ and $\mathsf{Exp}^{\mathrm{ideal}}_{\mathrm{fe}}$; otherwise, it is identical to Query I. (Note that the simulator could be required to decrypt $\mathsf{ct}' \neq \mathsf{ct}^*$ using $\mathsf{sk}_{j'}$ without knowing $C_{j'}(x^*)$.)
- $\mathcal{A}$ can submit some $j' \in [J]$ to retrieve a key, where J is the total number of key creation queries so far. If this is the first key retrieval query for j' in $\mathsf{Exp}^{\mathrm{ideal}}_{\mathrm{fe}}$, run

$$\mathsf{sk}_{j'} \xleftarrow{\$} \mathsf{Sim}(\text{“retrieve”}, j', C_{j'}(x^*)).$$

  In any case, $\mathsf{sk}_{j'}$ is now defined. Send $\mathsf{sk}_{j'}$ to $\mathcal{A}$.

- **Guess.** $\mathcal{A}$ outputs a bit β', which is the output of the experiment.

Remark 1. Security notions of FE can be compared typically along three axes: adaptivity of plaintext (x^*), IND/SIM, CPA/CCA. Simulation for fully adaptively chosen x^* is impossible [BSW11,AGVW13] in the standard model, even when the other two axes are IND-CPA. Definition 4 is otherwise *maximal* in strength. (The other maximal one is plaintext-adaptive IND-CCA.)

We did not find an existing definition of SIM-CCA for FE. Following [BL17], key creation and usage (retrieval/decryption) are separated.

Definition 5 (PKE, CCA). The definition of a *public-key encryption (PKE) scheme* is modified from FE scheme for circuits (Definition 3) by renaming Setup (resp. mpk, msk) to Gen (resp. pk, sk) and removing KeyGen and second-form Dec. The definition of *CCA(2) security* for PKE is modified from SIM-CCA for FE (Definition 4) by removing key creation/retrieval queries.

Ingredients of Construction 1. Let

- $\mathsf{PPRP} = (\mathsf{PPRP.Gen}, \mathsf{PPRP.Permute}, \mathsf{PPRP.Eval}, \mathsf{PPRP.Eval}^{-1})$ be an IP-PRP, and
- $i\mathcal{O}$ a circuit obfuscator.

Construction 1 (FE, simulator). The scheme works as follows.

- $\mathsf{Setup}(1^n)$ samples a PPRP key $\pi \xleftarrow{\$} \mathsf{PPRP.Gen}(1^{n+\lambda})$ and obfuscates the forward evaluation of π into $\widetilde{\pi} \xleftarrow{\$} i\mathcal{O}(\pi)$. The algorithm sets $\mathsf{mpk} \leftarrow \widetilde{\pi}$, $\mathsf{msk} \leftarrow \pi^{-1}$, and outputs $(\mathsf{mpk}, \mathsf{msk})$.
- $\mathsf{KeyGen}(\mathsf{msk}, C)$ runs and outputs $\mathsf{sk}_C \xleftarrow{\$} i\mathcal{O}(\mathsf{ct} \mapsto C(\mathsf{Dec}(\mathsf{msk}, \mathsf{ct})))$, where Dec is defined later.
- $\mathsf{Enc}(\mathsf{mpk}, x)$ parses mpk into $\widetilde{\pi}$, samples $r \xleftarrow{\$} \{0,1\}^\lambda$, and computes and outputs the ciphertext $\mathsf{ct} \leftarrow \widetilde{\pi}(x \| r)$.
- (1st form.) $\mathsf{Dec}(\mathsf{msk}, \mathsf{ct})$ parses msk into π^{-1}, computes $x \| r \leftarrow \pi^{-1}(\mathsf{ct})$, and outputs x (discarding the λ-bit r).
- (2nd form.) $\mathsf{Dec}(C, \mathsf{sk}_C, \mathsf{ct})$ runs and outputs $\mathsf{sk}_C(\mathsf{ct})$.

The correctness is readily verified. The simulator works as follows.

- $\mathsf{Sim}(\text{“setup”}, 1^n)$ runs

$$\pi \xleftarrow{\$} \mathsf{PPRP.Gen}(1^{n+\lambda}), \qquad \widetilde{\pi} \xleftarrow{\$} i\mathcal{O}(\pi), \qquad \mathsf{ct}^* \xleftarrow{\$} \{0,1\}^{n+\lambda},$$

  defines

$$R(y) := \begin{cases} \bot, & \text{if } y = \mathsf{ct}^*; \\ v, & \text{otherwise, } \pi^{-1}(y) = v \| r \text{ with } |r| = \lambda; \end{cases}$$

  and runs $\widetilde{R} \xleftarrow{\$} i\mathcal{O}(R)$. The algorithm initializes a list $\mathcal{L}$ of circuits with $\mathcal{L}[0] \leftarrow C_0$, where $C_0(x) := x$ is the identity circuit. It remembers $\widetilde{R}$, ct^*, $\mathcal{L}$, and outputs $\mathsf{mpk} \leftarrow \widetilde{\pi}$.

- $\mathsf{Sim}(\text{“create”}, C)$ adds C to the list so that $\mathcal{L}[J] = C$, where J is the length of $\mathcal{L}$ prior to appending.

- $\mathsf{Sim}(\text{“decrypt”}, j', \mathsf{ct}')$ reads $C_{j'} \leftarrow \mathcal{L}[j']$, and runs and outputs $C_{j'}(\widetilde{R}(\mathsf{ct}'))$, where Dec is the first form of decryption of the scheme. (If $\mathsf{ct}' = \mathsf{ct}^*$, i.e., if $\widetilde{R}(\mathsf{ct}') = \bot$, the response is set to $\bot$, even in Query I.)

- $\mathsf{Sim}(\text{“encrypt”})$ simply returns ct^*.

- $\mathsf{Sim}(\text{“retrieve”}, j', z^*_{j'})$ reads $C_{j'} \leftarrow \mathcal{L}[j']$ and defines

$$\widehat{C}_{j'}(\mathsf{ct}) := \begin{cases} z^*_{j'}, & \text{if } \mathsf{ct} = \mathsf{ct}^*; \\ C_{j'}(\widetilde{R}(\mathsf{ct})), & \text{otherwise.} \end{cases}$$

  It runs and outputs $\mathsf{sk}_{j'} \xleftarrow{\$} i\mathcal{O}(\widehat{C}_{j'})$.

All the circuits being obfuscated are appropriately padded.

Theorem 3 (FE; ¶). *Suppose $i\mathcal{O}$, PPRP are secure (Definitions 1 and 2) and PPRP supports all the permutations in Lemma 1, then Construction 1 is SIM-CCA-secure (Definition 4). Moreover, if the ingredients are subexponentially secure, then so is the scheme.*

The proof uses time-preserving reductions and $|\varepsilon_{\mathrm{fe}}| \le q/2^\lambda + J'_+|\varepsilon_{i\mathcal{O}}| + |\varepsilon_{\mathrm{cca}}|$, where q is the maximum number of decryption queries made by the adversary during Query I, J'_+ is a polynomial upper bound of the number of distinct key retrieval queries, and $\varepsilon_{\mathrm{cca}}$ is defined in Lemma 5 below.

Corollary 4 (PKE). *Suppose $i\mathcal{O}$, PPRP are secure (Definitions 1 and 2) and PPRP supports all the permutations in Lemma 1, then Construction 1 is CCA-secure as a PKE (Definition 5). Moreover, if the ingredients are subexponentially secure, then so is the scheme.*

The proof uses time-preserving reductions and $|\varepsilon_{\mathrm{pke}}| \le q/2^\lambda + |\varepsilon_{\mathrm{cca}}|$, where q is the maximum number of queries made by the adversary during Query I and $\varepsilon_{\mathrm{cca}}$ is defined in Lemma 5 below.

To prove the security of Construction 1, we rely on the following lemma: the challenge ciphertext remains pseudorandom in the presence of *obfuscations* of (not just oracle access to) punctured π^{-1}.

Lemma 5 (CCA; ¶). *Assuming the premise of Theorem 3, we have $\mathsf{Exp}^0_{\mathrm{cca}} \approx \mathsf{Exp}^1_{\mathrm{cca}}$, where $\mathsf{Exp}^\beta_{\mathrm{cca}}(\mathcal{A})$ proceeds as follows.*

- **Setup/Query I.** Launch $\mathcal{A}()$ and receive 1^n from it. Let

$$\pi \xleftarrow{\$} \mathsf{PPRP.Gen}(1^{n+\lambda}), \quad r^* \xleftarrow{\$} \{0,1\}^\lambda, \quad y_1 \xleftarrow{\$} \{0,1\}^{n+\lambda}, \quad \widetilde{P} \xleftarrow{\$} i\mathcal{O}(\pi),$$
$$Q_0 \leftarrow \pi^{-1} \setminus \left\{ \pi(v \| r^*) \,\middle|\, v \in \{0,1\}^n \right\}, \quad Q_1 \leftarrow \pi^{-1} \setminus \{y_1\}, \quad \widetilde{Q} \xleftarrow{\$} i\mathcal{O}(Q_\beta),$$

  Send $(\widetilde{P}, \widetilde{Q})$ to $\mathcal{A}$. (Recall the notation — Q_0, Q_1 are punctured versions of π^{-1}. The puncturing in Q_0 uses π^{-1}, as detailed in G_0 in the proof.)

- **Challenge/Query II.** $\mathcal{A}$ chooses $v^* \in \{0,1\}^n$. Set $y_0 \leftarrow \pi(v^* \| r^*)$, $y^* \leftarrow y_\beta$. Define

$$R(y) := \begin{cases} \perp, & \text{if } y = y^*; \\ v, & \text{otherwise, } \pi^{-1}(y) = v \| r \text{ with } |r| = \lambda. \end{cases}$$

  Run $\widetilde{R} \xleftarrow{\$} i\mathcal{O}(R)$ and send $(y^*, \widetilde{R})$ to $\mathcal{A}$.

- **Guess.** $\mathcal{A}$ outputs a bit β', which is the output of the experiment.

Circuits being obfuscated are appropriately padded. The proof uses time-preserving reductions and the advantage relation is $\varepsilon_{\mathrm{cca}} = 22\varepsilon_{i\mathcal{O}} + 4\varepsilon_{\mathrm{trigger}} + 2\varepsilon_{\mathsf{PPRP}}$.

We discuss abstractions related to Lemma 5 in Section 7.1. Before going into its proof, let's see how Lemma 5 is used.

Proof (Theorem 3). We go through the following hybrids, each described by changes made to the previous one.

- H_0 is $\mathsf{Exp}^{\mathrm{real}}_{\mathrm{fe}}$ with the following changes: the encryption randomness $r^* \xleftarrow{\$} \{0,1\}^\lambda$ for ct^* is sampled at the very beginning; a decryption query to ct' using key j' is answered as $C_{j'}(\mathsf{Dec}(\mathsf{msk}, \mathsf{ct}'))$, where we supply the definition of $C_0(x) := x$; a secret key $\mathsf{sk}_{j'}$ is generated upon its first retrieval query.

  These changes are only conceptual, so $\mathsf{Exp}^{\mathrm{real}}_{\mathrm{fe}} \equiv \mathsf{H}_0$.

- In H_1, a decryption query in Query I is responded with $\perp$ if $\pi^{-1}(\mathsf{ct}') = \cdots \| r^*$ (i.e., if the encryption randomness in ct' collides with r^*, the challenge one). Query II is unchanged (rejecting exactly ct^*).

  In H_0, the adversary's view is independent of r^* prior to Challenge, so (the "bad event")

$$\Pr\left[\pi^{-1}(\mathsf{ct}') = \cdots \| r^* \text{ for some } \mathsf{ct}' \text{ during Query I} \right] \le q/2^\lambda = \mathrm{negl}(\lambda),$$

  where q is the maximum number of decryption queries in Query I. The two hybrids differ only when the bad event happens, hence $\mathsf{H}_0 \approx_{\mathrm{s}} \mathsf{H}_1$.

- In H_2, the secret keys are generated in a different manner. When x^* is chosen by the adversary, we compute $\mathsf{ct}^* \leftarrow \pi(x^* \| r^*)$ as required, define

$$R(y) := \begin{cases} \perp, & \text{if } y = \mathsf{ct}^*; \\ v, & \text{otherwise, } \pi^{-1}(y) = v \| r \text{ with } |r| = \lambda; \end{cases}$$

and generate $\widetilde{R} \xleftarrow{\$} i\mathcal{O}(R)$. Upon the first key retrieval query for $\mathsf{sk}_{j'}$, compute $z^*_{j'} = C_{j'}(x^*)$, define

$$\widehat{C}_{j'}(\mathsf{ct}) := \begin{cases} z^*_{j'}, & \text{if } \mathsf{ct} = \mathsf{ct}^*; \\ C_{j'}(\widetilde{R}(\mathsf{ct})), & \text{otherwise;} \end{cases}$$

and let $\mathsf{sk}_{j'} \xleftarrow{\$} i\mathcal{O}(\widehat{C}_{j'})$. (Note that H_2 is syntactically valid, as key retrieval queries are allowed only after x^* is chosen, at which point $\widehat{C}_{j'}$ can be written with ct^*, R properly defined.)

Because the functionalities of $\mathsf{sk}_{j'}$ are unchanged, $\mathsf{H}_1 \approx \mathsf{H}_2$ by $i\mathcal{O}$ security.

- In H_3, we make the following changes: ct^* is sampled as $\mathsf{ct}^* \xleftarrow{\$} \{0,1\}^{n+\lambda}$ at the beginning; a decryption query in Query I is rejected if and only if $\mathsf{ct}' = \mathsf{ct}^*$, using the same rule as in Query II. (The change of ct^* naturally propagates into R and hence $\mathsf{sk}_{j'}$'s.)

  $\mathsf{H}_2 \approx \mathsf{H}_3$ reduces to Lemma 5 as follows. The reduction algorithm forwards the choice of 1^n, sets $\mathsf{mpk} \leftarrow \widetilde{P}$, and uses $\widetilde{Q}$ to answer Query I decryption queries. It forwards the choice of x^*, sets $\mathsf{ct}^* \leftarrow y^*$, and handles Query II (decryption and key retrieval) using $\widetilde{R}$. Eventually, the reduction outputs the bit produced by the adversary.

Note that $\mathsf{H}_3 \equiv \mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{fe}}$. The proof completes by hybrid argument. □

Remark 2. If one wants to prove Corollary 4 directly instead of obtaining it from Theorem 3... First, there are no direct reductions to $i\mathcal{O}$ security as no key is being retrieved, matching the fact that $\varepsilon_{i\mathcal{O}}$ does not appear directly in the advantage (i.e., $J'_+ = 0$). Second, Lemma 5 is an overkill, in the sense that an alternative proof can be completed without using $\widetilde{R}$ (call the version of lemma without $\widetilde{R}$ "weak"). An alternative proof, between H_1 and H_3, first switches Query II to be handled using $\widetilde{Q}$, justified by bounding the bad event probability by reduction to the weak lemma. It concludes by another reduction to the weak lemma to replace ct^* by random. (See also Section 7.1.)

In contrast, for FE, both $\widetilde{Q}$ and $\widetilde{R}$ (the full power of Lemma 5) are needed. We rely on $\widetilde{Q}$ to handle pre-challenge queries and we can create functional keys only with $\widetilde{R}$ (not with $\widetilde{Q}$, as it would be unclear how to argue indistinguishability from honest keys due to over-puncturing).

Proof (Lemma 5). Let $(\not{\Delta}, \Delta)$ be a trigger scheme[10] promised by Lemma 2. Let

$$\begin{aligned} u^* &\xleftarrow{\$} \{0,1\}^n, & s^*_1 &\xleftarrow{\$} \{0,1\}^\lambda, & \Gamma_1 &\leftarrow \mathsf{URS}[u^*, r^*, s^*_1], \\ w^* &\xleftarrow{\$} \{0,1\}^n, & s^*_2 &\xleftarrow{\$} \{0,1\}^\lambda, & \Gamma_2 &\leftarrow (w^* \| s^*_1 \quad w^* \| s^*_2), \end{aligned}$$

none used in $\mathsf{Exp}^0_{\mathsf{cca}}$, $\mathsf{Exp}^1_{\mathsf{cca}}$. Recall (from Lemma 1) that Γ_1 is the permutation

$$\Gamma_1(v\|r) = \begin{cases} (u^* \oplus v)\|s^*_1, & \text{if } r = r^*; \\ v \ \|r^*, & \text{if } r = s^*_1 \text{ and } r \neq r^*; \\ v \ \|r \ , & \text{otherwise.} \end{cases}$$

[10] When PPRP, $i\mathcal{O}$ are subexponentially secure, we can and will use a subexponentially secure trigger scheme. Otherwise, it can be just polynomially secure.

We will first permute π by Γ_1. This changes y^* to $\pi((u^* \oplus v^*)\|s_1^*)$, which almost does the job, but R will leak u^* (thus v^*) when its input is not y^* yet happens to have s_1^* as suffix of preimage under π. Moreover, s_1^* appears in y^*, so it cannot be removed (yet) from R as a random trigger. Of course, the intuition is that the leaky branch cannot be detected by the adversary. To argue this, we rename $(u^* \oplus v^*)$ to w^*, then permute π again, this time by Γ_2, taking s_1^* away from y^*. After that, we can delete s_1^* as a trigger, which fully erases information about v^*.

Denote by P, Q the programs being obfuscated into $\widetilde{P}$, $\widetilde{Q}$, and by $\mathcal{T}_0$, $\mathcal{T}_1$, $\mathcal{T}_2$ the triggers potentially activated at r^*, s_1^*, s_2^*, respectively. Formally, we go through the following hybrids, which are fully specified by P, Q, y^*, R.

- G_0 is $\mathsf{Exp}^0_{\mathsf{cca}}$, where

$$P(v\|r) = \pi(v\|r), \qquad Q(y) = \begin{cases} \bot, & \text{if } \pi^{-1}(y) = v\|r \text{ with } r = r^*; \\ \pi^{-1}(y), & \text{otherwise;} \end{cases}$$
$$y^* = \pi(v^*\|r^*), \qquad R(y) = \begin{cases} \bot, & \text{if } y = y^*; \\ v, & \text{otherwise, } \pi^{-1}(y) = v\|r \text{ with } |r| = \lambda. \end{cases}$$

- In G_1, we sample $\mathcal{T}_1 \xleftarrow{\$} \mathbb{\Delta}$ and change the programs:[11]

$$P(v\|r) = \begin{cases} \boxed{\pi((u^* \oplus v)\|r^*),} & \boxed{\text{if } \mathcal{T}_1(r) = 1;} \\ \pi(v\|r), & \text{otherwise;} \end{cases}$$
$$Q(y) = \begin{cases} \bot, & \text{if } \pi^{-1}(y) = v\|r \text{ with } r = r^* \boxed{\text{ or } \mathcal{T}_1(r) = 1}; \\ \pi^{-1}(y), & \text{otherwise;} \end{cases}$$
$$y^* = \pi(v^*\|r^*),$$
$$R(y) = \begin{cases} \bot, & \text{if } y = y^*; \\ v, & \text{otherwise, } \pi^{-1}(y) = v\|r \text{ with } |r| = \lambda. \end{cases}$$

  The newly added branches are never entered, so $\mathsf{G}_0 \approx \mathsf{G}_1$ by $i\mathcal{O}$ security.

- In G_2, we change $\mathcal{T}_1$ to $\mathcal{T}_1 \xleftarrow{\$} \boxed{\Delta(s_1^*)}$.

  By trigger scheme security, $\mathsf{G}_1 \approx \mathsf{G}_2$.

- In G_3, we run $\pi_* \xleftarrow{\$} \mathsf{PPRP.Permute}(\pi, \Gamma_1, 0)$ and update the programs and y^* to use s_1^*, π_*:

$$P(v\|r) = \begin{cases} \boxed{\pi_*}((u^* \oplus v)\|r^*), & \text{if } \boxed{r = s_1^*}; \\ \boxed{\pi_*}(v\|r), & \text{otherwise;} \end{cases}$$
$$Q(y) = \begin{cases} \bot, & \text{if } \boxed{\pi_*^{-1}}(y) = v\|r \text{ with } r = r^* \text{ or } \boxed{r = s_1^*}; \\ \boxed{\pi_*^{-1}}(y), & \text{otherwise;} \end{cases}$$

[11] Note that $(u^* \oplus v)\|r^* = \Gamma_1^{-1}(v\|s_1^*)$, following the usual trick of composing Γ_1^{-1} so that the triggered branch will revert to normal once the PPRP key is permuted by Γ_1.

$$y^* = \boxed{\pi_*}(v^* \| r^*),$$
$$R(y) = \begin{cases} \bot, & \text{if } y = y^*; \\ v, & \text{otherwise, } \boxed{\pi_*^{-1}}(y) = v\|r \text{ with } |r| = \lambda. \end{cases}$$

By the correctness of PPRP and the trigger scheme, y^* and the functionalities of P, Q, R are unchanged. Therefore, $\mathsf{G}_2 \approx \mathsf{G}_3$ by $i\mathcal{O}$ security.

- In G_4, we replace π_* by $\pi_* \xleftarrow{\$} \mathsf{PPRP.Permute}(\pi, \Gamma_1, \boxed{1})$.

  By PPRP security, $\mathsf{G}_3 \approx \mathsf{G}_4$.

- In G_5, we sample $\mathcal{T}_0 \xleftarrow{\$} \Delta(r^*)$ and rewrite all the components to use π, $\mathcal{T}_0$ instead of π_*, r^*:

$$\boxed{P}(v\|r) = \begin{cases} \pi((u^* \oplus v)\|s_1^*), & \text{if } \boxed{\mathcal{T}_0(r) = 1} \text{ and } r \neq s_1^*; \\ \pi(v\|r), & \text{otherwise;} \end{cases}$$
$$\boxed{Q}(y) = \begin{cases} \bot, & \text{if } \pi^{-1}(y) = v\|r \text{ with } r = s_1^* \text{ or } \boxed{\mathcal{T}_0(r) = 1}; \\ \pi^{-1}(y), & \text{otherwise;} \end{cases}$$
$$y^* = \boxed{\pi((u^* \oplus v^*)\|s_1^*)},$$
$$\boxed{R}(y) = \begin{cases} \bot, & \text{if } y = y^*; \\ v \oplus u^*, & \text{else if } \pi^{-1}(y) = v\|r \text{ with } r = s_1^*; \\ v, & \text{otherwise, } \pi^{-1}(y) = v\|r \text{ with } |r| = \lambda. \end{cases}$$

  By the correctness of PPRP and the trigger scheme, y^* and the functionalities of P, Q, R are unchanged. Therefore, $\mathsf{G}_4 \approx \mathsf{G}_5$ by $i\mathcal{O}$ security.

- In G_6, we switch $\mathcal{T}_0$ to $\mathcal{T}_0 \xleftarrow{\$} \boxed{\not\Delta}$.

  By trigger scheme security, $\mathsf{G}_5 \approx \mathsf{G}_6$.

- In G_7, we remove the inactive branches:

$$P(v\|r) = \begin{cases} \sout{\pi((u^* \oplus v)\|s_1^*),} & \sout{\text{if } \mathcal{T}_0(r) = 1 \text{ and } r \neq s_1^*;} \\ \pi(v\|r), & \sout{\text{otherwise;}} \end{cases}$$
$$Q(y) = \begin{cases} \bot, & \text{if } \pi^{-1}(y) = v\|r \text{ with } r = s_1^* \sout{\text{ or } \mathcal{T}_0(r) = 1}; \\ \pi^{-1}(y), & \text{otherwise;} \end{cases}$$
$$y^* = \pi((u^* \oplus v^*)\|s_1^*),$$
$$R(y) = \begin{cases} \bot, & \text{if } y = y^*; \\ v \oplus u^*, & \text{else if } \pi^{-1}(y) = v\|r \text{ with } r = s_1^*; \\ v, & \text{otherwise, } \pi^{-1}(y) = v\|r \text{ with } |r| = \lambda. \end{cases}$$

  By $i\mathcal{O}$ security, $\mathsf{G}_6 \approx \mathsf{G}_7$. (This removes u^* from P.)

- In G_8, we rename $(u^* \oplus v^*)$ to w^*. The components become

$$P(v\|r) = \pi(v\|r),$$

$$Q(y) = \begin{cases} \bot, & \text{if } \pi^{-1}(y) = v\|r \text{ with } r = s_1^*; \\ \pi^{-1}(y), & \text{otherwise;} \end{cases}$$

$$y^* = \pi(\boxed{w^*}\|s_1^*),$$

$$R(y) = \begin{cases} \bot, & \text{if } y = y^*; \\ v \oplus \boxed{(v^* \oplus w^*)}, & \text{else if } \pi^{-1}(y) = v\|r \text{ with } r = s_1^*; \\ v, & \text{otherwise, } \pi^{-1}(y) = v\|r \text{ with } |r| = \lambda. \end{cases}$$

Above and hereafter, $(v^* \oplus w^*)$ is written in R as a value (if appearing at all), not as some computation using v^*, w^*.

The change of variable is conceptual, so $\mathsf{G}_7 \equiv \mathsf{G}_8$. (We note that removing u^* from P in G_7 is crucial, as otherwise, v^*, adaptively chosen from P, would have to be (circularly!) embedded into P.)

- In G_9, we sample $\mathcal{T}_2 \xleftarrow{\$} \not\Delta$ and rewrite the programs:

$$P(v\|r) = \begin{cases} \boxed{\pi(w^*\|s_1^*),} & \boxed{\text{if } v\|\mathcal{T}_2(r) = w^*\|1;} \\ \pi(v\|r), & \text{otherwise;} \end{cases}$$

$$Q(y) = \begin{cases} \bot, & \text{if } \pi^{-1}(y) = v\|r \text{ with } r = s_1^* \boxed{\text{ or } v\|\mathcal{T}_2(r) = w^*\|1}; \\ \pi^{-1}(y), & \text{otherwise;} \end{cases}$$

$$y^* = \pi(w^*\|s_1^*),$$

$$R(y) = \begin{cases} \bot, & \text{if } y = y^* \boxed{\text{ or } \pi^{-1}(y) = v\|r \text{ with } v\|\mathcal{T}_2(r) = w^*\|1}; \\ v \oplus (v^* \oplus w^*), & \text{else if } \pi^{-1}(y) = v\|r \text{ with } r = s_1^*; \\ v, & \text{otherwise, } \pi^{-1}(y) = v\|r \text{ with } |r| = \lambda. \end{cases}$$

The newly added branches are never entered, so $\mathsf{G}_8 \approx \mathsf{G}_9$ by $i\mathcal{O}$ security.

- In G_{10}, we change $\mathcal{T}_2$ to $\mathcal{T}_2 \xleftarrow{\$} \boxed{\Delta(s_2^*)}$.

By trigger scheme security, $\mathsf{G}_9 \approx \mathsf{G}_{10}$.

- In G_{11}, we run $\pi_* \xleftarrow{\$} \mathsf{PPRP.Permute}(\pi, \Gamma_2, 0)$ and update the programs and y^* to use π_*, s_2^*:

$$P(v\|r) = \begin{cases} \boxed{\pi_*}(w^*\|s_1^*), & \text{if } \boxed{v\|r = w^*\|s_2^*}; \\ \boxed{\pi_*}(v\|r), & \text{otherwise;} \end{cases}$$

$$Q(y) = \begin{cases} \bot, & \text{if } \boxed{\pi_*^{-1}}(y) = v\|r \text{ with } r = s_1^* \text{ or } \boxed{v\|r = w^*\|s_2^*}; \\ \boxed{\pi_*^{-1}}(y), & \text{otherwise;} \end{cases}$$

$$y^* = \boxed{\pi_*}(w^*\|s_1^*),$$

$$R(y) = \begin{cases} \bot, & \text{if } \boxed{\pi_*^{-1}(y) = w^*\|s_1^* \text{ or } \pi_*^{-1}(y) = w^*\|s_2^*}; \\ v \oplus (v^* \oplus w^*), & \text{else if } \boxed{\pi_*^{-1}}(y) = v\|r \text{ with } r = s_1^*; \\ v, & \text{otherwise, } \boxed{\pi_*^{-1}}(y) = v\|r \text{ with } |r| = \lambda. \end{cases}$$

By the correctness of PPRP and the trigger scheme, y^* and the functionalities of P, Q, R are unchanged. We have $\mathsf{G}_{10} \approx \mathsf{G}_{11}$ by $i\mathcal{O}$ security.

- In G_{12}, we replace π_* by $\pi_* \xleftarrow{\$} \mathsf{PPRP.Permute}(\pi, \Gamma_2, 1)$.

  By PPRP security, $\mathsf{G}_{11} \approx \mathsf{G}_{12}$. (Recall that Γ_2 is $(w^* \| s_1^* \quad w^* \| s_2^*)$.)

- In G_{13}, we sample $\mathcal{T}_1 \xleftarrow{\$} \Delta(s_1^*)$ and rewrite everything using π, $\mathcal{T}_1$ instead of π_*, s_1^*. The components become

$$P(v\|r) = \begin{cases} \pi(w^*\|s_2^*), & \text{if } v\|\mathcal{T}_1(r) = w^*\|1; \\ \pi(v\|r), & \text{otherwise;} \end{cases}$$

$$Q(y) = \begin{cases} \bot, & \text{if } y = y^* \text{ or } \pi^{-1}(y) = v\|r \text{ with } \mathcal{T}_1(r) = 1; \\ \pi^{-1}(y), & \text{otherwise;} \end{cases}$$

$$y^* = \pi(w^*\|s_2^*),$$

$$R(y) = \begin{cases} \bot, & \text{if } y = y^* \text{ or } \pi^{-1}(y) = v\|r \text{ with } v\|\mathcal{T}_1(r) = w^*\|1; \\ v \oplus (v^* \oplus w^*), & \text{else if } \pi^{-1}(y) = v\|r \text{ with } \mathcal{T}_1(r) = 1; \\ v, & \text{otherwise, } \pi^{-1}(y) = v\|r \text{ with } |r| = \lambda. \end{cases}$$

  By the correctness of PPRP and the trigger scheme, y^* and the functionalities of P, Q, R are unchanged. We have $\mathsf{G}_{12} \approx \mathsf{G}_{13}$ by $i\mathcal{O}$ security.

- In G_{14}, we switch $\mathcal{T}_1$ to $\mathcal{T}_1 \xleftarrow{\$} \not{\Delta}$.

  By trigger scheme security, $\mathsf{G}_{13} \approx \mathsf{G}_{14}$.

- In G_{15}, we clean up the components:

$$P(v\|r) = \begin{cases} \cancel{\pi(w^*\|s_2^*),} & \cancel{\text{if } v\|\mathcal{T}_1(r) = w^*\|1;} \\ \pi(v\|r), & \cancel{\text{otherwise;}} \end{cases}$$

$$Q(y) = \begin{cases} \bot, & \text{if } y = y^* \ \cancel{\text{or } \pi^{-1}(y) = v\|r \text{ with } \mathcal{T}_1(r) = 1}; \\ \pi^{-1}(y), & \text{otherwise;} \end{cases}$$

$$y^* = \pi(w^*\|s_2^*),$$

$$R(y) = \begin{cases} \bot, & \text{if } y = y^* \ \cancel{\text{or } \pi^{-1}(y) = v\|r \text{ with } v\|\mathcal{T}_1(r) = w^*\|1}; \\ \cancel{v \oplus (v^* \oplus w^*),} & \cancel{\text{else if } \pi^{-1}(y) = v\|r \text{ with } \mathcal{T}_1(r) = 1;} \\ v, & \text{otherwise, } \pi^{-1}(y) = v\|r \text{ with } |r| = \lambda. \end{cases}$$

  By $i\mathcal{O}$ security, $\mathsf{G}_{14} \approx \mathsf{G}_{15}$. (This finally removes v^* from R, hence everywhere in the hybrid.) Note that $\mathsf{G}_{15} \equiv \mathsf{Exp}^1_{\mathsf{cca}}$ by renaming $\pi(w^*\|s_2^*)$ to y_1.

The proof completes by hybrid argument. □

5 Key Encapsulation Mechanism

Although any CCA-secure PKE can be used as a key encapsulation mechanism, it is suboptimal since a ciphertext is longer than the entropy of an encapsulated key. Note that keys can be stretched using a PRG, so we use true entropy as the discriminating measure.

We now construct a KEM where an encapsulated key has exactly $|\mathsf{ct}|$ bits of entropy, i.e., maximally size-efficient. (Moreover, its computational security is effectively only limited by that of triggers. See Remark 3.) The idea is to use *two* PPRP instances.

We first recall the definition.

Definition 6 (KEM, CCA). A *key encapsulation mechanism (KEM)* consists of three efficient algorithms.

- $\mathsf{Gen}(1^\lambda)$ outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Enc}(1^\lambda, \mathsf{pk})$ takes pk as input. It outputs a ciphertext ct and an encapsulated key $k \in \{0,1\}^\lambda$.
- $\mathsf{Dec}(1^\lambda, \mathsf{sk}, \mathsf{ct})$ takes as input pk, ct. It is supposed to recover k.

The scheme must be *correct*, i.e., for all $\lambda \in \mathbb{N}$, it holds that

$$\Pr\left[\begin{array}{l}(\mathsf{pk},\mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda) \\ (\mathsf{ct}, k) \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{pk})\end{array} : \mathsf{Dec}(1^\lambda, \mathsf{sk}, \mathsf{ct}) = k\right] = 1.$$

The scheme is *CCA-secure* if $\mathsf{Exp}^0_{\mathrm{kem}} \approx \mathsf{Exp}^1_{\mathrm{kem}}$, where $\mathsf{Exp}^\beta_{\mathrm{kem}}(1^\lambda, \mathcal{A})$ proceeds as follows.

- **Challenge.** Run

  $$(\mathsf{pk},\mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda), \quad (\mathsf{ct}^*, k_0) \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{pk}), \quad k_1 \xleftarrow{\$} \{0,1\}^\lambda, \quad k^* \leftarrow k_\beta.$$

  Launch $\mathcal{A}(1^\lambda)$ and send $(\mathsf{pk}, \mathsf{ct}^*, k^*)$ to it.
- **Query.** Repeat the following for arbitrarily many rounds determined by $\mathcal{A}$. In each round, $\mathcal{A}$ submits some ct for decryption. Upon such a query, if $\mathsf{ct} = \mathsf{ct}^*$, send $\perp$ to $\mathcal{A}$. Otherwise, run $\mathsf{Dec}(1^\lambda, \mathsf{sk}, \mathsf{ct})$ and send its output to $\mathcal{A}$.
- **Guess.** $\mathcal{A}$ outputs a bit β', which is the output of the experiment.

Ingredients of Construction 2. Let

- $\mathsf{PPRP} = (\mathsf{PPRP.Gen}, \mathsf{PPRP.Permute}, \mathsf{PPRP.Eval}, \mathsf{PPRP.Eval}^{-1})$ be an IP-PRP, and
- $i\mathcal{O}$ a circuit obfuscator.

Construction 2 (KEM). It works as follows.

- $\mathsf{Gen}()$ runs[12] (with appropriate padding in calls to $i\mathcal{O}$)

  $$\pi \xleftarrow{\$} \mathsf{PPRP.Gen}(1^\lambda), \quad \widetilde{\pi} \xleftarrow{\$} i\mathcal{O}(\pi), \quad \sigma \xleftarrow{\$} \mathsf{PPRP.Gen}(1^\lambda), \quad \widetilde{\sigma} \xleftarrow{\$} i\mathcal{O}(\sigma),$$

  sets $\mathsf{pk} \leftarrow (\widetilde{\pi}, \widetilde{\sigma})$ and $\mathsf{sk} \leftarrow (\pi^{-1}, \sigma)$, and outputs $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Enc}(\mathsf{pk})$ first parses pk into $(\widetilde{\pi}, \widetilde{\sigma})$. It then samples $r \xleftarrow{\$} \{0,1\}^\lambda$ and computes $\mathsf{ct} \leftarrow \widetilde{\pi}(r)$ and $k \leftarrow \widetilde{\sigma}(r)$. The algorithm outputs (ct, k).
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ parses sk into (π^{-1}, σ). It computes and outputs $k \leftarrow \sigma(\pi^{-1}(\mathsf{ct}))$.

The scheme is clearly correct.

[12] The argument 1^λ to $\mathsf{PPRP.Gen}$ indicates the domain size. The security parameter is still implicitly λ.

Theorem 6 (¶). *Suppose $i\mathcal{O}$, PPRP are secure (Definitions 1 and 2) and PPRP supports all transpositions (see Lemma 1), then Construction 2 is CCA-secure (Definition 6). Moreover, if the ingredients are subexponentially secure, then so is the scheme.*

The proof uses time-preserving reductions and the advantage relation is $\varepsilon_{\text{kem}} = \varepsilon_{\text{cca-rr}}$, where $\varepsilon_{\text{cca-rr}}$ is defined in Lemma 7 below.

To prove the security of Construction 2, we rely on the following stronger lemma, which will also be used in Section 6 (unary HE).

Lemma 7 (plaintext-selective CCA with rerandomization by PPRP; ¶). *Let $i\mathcal{O}$ be an indistinguishability obfuscator and $\mathsf{PPRP} = (\mathsf{Gen}, \mathsf{Permute}, \mathsf{Eval}, \mathsf{Eval}^{-1})$ an IP-PRP supporting all transpositions (Definitions 1 and 2 and Lemma 1), then $\mathsf{Exp}^0_{\text{cca-rr}} \approx \mathsf{Exp}^1_{\text{cca-rr}}$, where $\mathsf{Exp}^\beta_{\text{cca-rr}}(\mathcal{A})$ proceeds as follows.*

- **Challenge.** Launch $\mathcal{A}()$ and receive $v^* \in \{0,1\}^\star$ from it. Run

$$\begin{aligned}
&\pi \xleftarrow{\$} \mathsf{Gen}(1^{|v^*|+\lambda}), && \sigma \xleftarrow{\$} \mathsf{Gen}(1^\lambda), && r^* \xleftarrow{\$} \{0,1\}^\lambda,\\
&y_0 \leftarrow \pi(v^* \| r^*), && y_1 \xleftarrow{\$} \{0,1\}^{|v^*|+\lambda}, && y^* \leftarrow y_\beta,\\
&s_0 \leftarrow \sigma(r^*), && s_1 \xleftarrow{\$} \{0,1\}^\lambda, && s^* \leftarrow s_\beta,\\
&\widetilde{P} \xleftarrow{\$} i\mathcal{O}(\pi), && \widetilde{Q} \xleftarrow{\$} i\mathcal{O}(\sigma), && \widetilde{R} \xleftarrow{\$} i\mathcal{O}(\pi^{-1} \setminus \{y^*\}),
\end{aligned}$$

  where the calls to $i\mathcal{O}$ are appropriately padded. Send $(y^*, s^*, \widetilde{P}, \widetilde{Q}, \widetilde{R})$ to $\mathcal{A}$.

- **Guess.** $\mathcal{A}$ outputs a bit β', which is the output of the experiment.

The proof uses time-preserving reductions and $\varepsilon_{\text{cca-rr}} = 11\varepsilon_{i\mathcal{O}} + 3\varepsilon_{\text{trigger}} + 2\varepsilon_{\mathsf{PPRP}}$.

The lemma can be further generalized. (See Section 7.2.)

Proof (Theorem 6). The reduction to Lemma 7 simply sets v^* to be the empty string, $\mathsf{pk} \leftarrow (\widetilde{P}, \widetilde{Q})$, $\mathsf{ct}^* \leftarrow y^*$, $k^* \leftarrow s^*$, and answers decryption queries by $\widetilde{Q}(\widetilde{R}(\mathsf{ct}))$. It outputs the bit produced by the adversary when the latter eventually halts. □

Remark 3. When $i\mathcal{O}$ and PPRP are subexponentially secure, their security parameters can be dialed up so that the (computational) security of KEM attains the maximum possible for λ-bit triggers (modulo a constant reduction loss) while maintaining $|\mathsf{ct}| = |k| = \lambda$. This complements the "statistical" efficiency (k has $|\mathsf{ct}|$ bits of entropy). To maximize the security of triggers, they should be constructed more cleverly. The route in Section A aims to minimize the kinds of assumptions needed for the overall paper, and its known proof is highly non-tight.

Proof (Lemma 7). Sample $x' \xleftarrow{\$} \{0,1\}^{|v^*|+\lambda}$, $r' \xleftarrow{\$} \{0,1\}^\lambda$. Let $(\not{\Delta}, \Delta)$ be a trigger scheme (potentially subexponentially secure) promised by Lemma 2 and $\mathcal{T}_0$, $\mathcal{T}_1$, $\mathcal{T}_2$ triggers potentially activated at r^*, x', r'.

Observe that y_0, s_0 are correlated by their preimages under π, σ, and security translates to them being computationally jointly uniform (note that neither preimage is accessible to the adversary). The crux of the proof is to permute π by $(v^* \| r^* \;\; x')$ and σ by $(r^* \;\; r')$, which converts those preimages to x', r', i.e., following the desired distribution.

Writing P, Q, R for the programs being obfuscated into $\widetilde{P}$, $\widetilde{Q}$, $\widetilde{R}$, we consider the following hybrids, which are fully specified by y^*, s^*, P, Q, R.

- G_0 is $\mathsf{Exp}^0_{\text{cca-rr}}$, where

$$y^* = \pi(v^* \| r^*), \qquad s^* = \sigma(r^*),$$
$$P(v\|r) = \pi(v\|r), \qquad Q(r) = \sigma(r),$$
$$R(y) = \begin{cases} \bot, & \text{if } y = y^*; \\ \pi^{-1}(y), & \text{otherwise.} \end{cases}$$

- In G_1, we sample $\mathcal{T}_1 \xleftarrow{\$} \mathbb{\Lambda}_{|v^*|+\lambda}$ and $\mathcal{T}_2 \xleftarrow{\$} \mathbb{\Lambda}_\lambda$. The programs are rewritten:

$$y^* = \pi(v^* \| r^*), \qquad s^* = \sigma(r^*),$$
$$P(v\|r) = \begin{cases} \boxed{\pi(v^*\|r^*),} & \boxed{\text{if } \mathcal{T}_1(v\|r) = 1;} \\ \pi(v\|r), & \text{otherwise;} \end{cases} \qquad Q(r) = \begin{cases} \boxed{\sigma(r^*),} & \boxed{\text{if } \mathcal{T}_2(r) = 1;} \\ \sigma(r), & \text{otherwise;} \end{cases}$$
$$R(y) = \begin{cases} \bot, & \text{if } y = y^*; \\ \boxed{v^*\|r^*,} & \boxed{\text{else if } \mathcal{T}_1(\pi^{-1}(y)) = 1;} \\ \pi^{-1}(y), & \text{otherwise.} \end{cases}$$

  Since the newly added branches are never entered, $\mathsf{G}_0 \approx \mathsf{G}_1$ by $i\mathcal{O}$ security.

- In G_2, we activate the triggers by $\mathcal{T}_1 \xleftarrow{\$} \boxed{\Delta(x')}$ and $\mathcal{T}_2 \xleftarrow{\$} \boxed{\Delta(r')}$.

  By trigger scheme security, we have $\mathsf{G}_1 \approx \mathsf{G}_2$.

- In G_3, we run

$$\pi_* \xleftarrow{\$} \mathsf{PPRP.Permute}(\pi, (v^*\|r^* \;\; x'), 0), \qquad \sigma_* \xleftarrow{\$} \mathsf{PPRP.Permute}(\sigma, (r^* \;\; r'), 0),$$

  and rewrite the components to use π_*, σ_*, x', r':

$$y^* = \boxed{\pi_*}(v^*\|r^*), \qquad s^* = \boxed{\sigma_*}(r^*),$$
$$P(v\|r) = \begin{cases} \boxed{\pi_*}(v^*\|r^*), & \text{if } \boxed{v\|r = x'}; \\ \boxed{\pi_*}(v\|r), & \text{otherwise;} \end{cases} \qquad Q(r) = \begin{cases} \boxed{\sigma_*}(r^*), & \text{if } \boxed{r = r'}; \\ \boxed{\sigma_*}(r), & \text{otherwise;} \end{cases}$$
$$R(y) = \begin{cases} \bot, & \text{if } \boxed{\pi_*^{-1}(y) = v^*\|r^*}; \\ v^*\|r^*, & \text{else if } \boxed{\pi_*^{-1}(y) = x'}; \\ \boxed{\pi_*^{-1}}(y), & \text{otherwise.} \end{cases}$$

  By the correctness of PPRP and trigger scheme, y^*, s^*, the functionalities of P, Q, R are unchanged, and $\mathsf{G}_2 \approx \mathsf{G}_3$ by $i\mathcal{O}$ security.

- In G_4, we switch π_*, σ_* to permuted by running

$$\pi_* \xleftarrow{\$} \mathsf{PPRP.Permute}(\pi, (v^*\|r^* \;\; x'), \boxed{1}), \qquad \sigma_* \xleftarrow{\$} \mathsf{PPRP.Permute}(\sigma, (r^* \;\; r'), \boxed{1}).$$

  We have $\mathsf{G}_3 \approx \mathsf{G}_4$ by PPRP security.

- In G_5, we sample $\mathcal{T}_0 \xleftarrow{\$} \Delta(r^*)$ and produce everything using π, σ, $\mathcal{T}_0$ instead of the permuted keys and r^*:

$$y^* = \boxed{\pi(x')}, \qquad s^* = \boxed{\sigma(r')},$$
$$\boxed{P}(v\|r) = \begin{cases} \pi(x'), & \text{if } \boxed{v\|\mathcal{T}_0(r) = v^*\|1}; \\ \pi(v\|r), & \text{otherwise;} \end{cases} \qquad \boxed{Q}(r) = \begin{cases} \sigma(r'), & \text{if } \boxed{\mathcal{T}_0(r) = 1}; \\ \sigma(r), & \text{otherwise;} \end{cases}$$

$$R(y) = \begin{cases} \bot, & \text{if } y = y^*; \\ \pi^{-1}(y), & \text{otherwise.} \end{cases}$$

By the correctness of PPRP and trigger scheme, y^*, s^*, the functionalities of P, Q, R are unchanged, so $\mathsf{G}_4 \approx \mathsf{G}_5$ by $i\mathcal{O}$ security.

- In G_6, we deactivate the trigger by $\mathcal{T}_0 \xleftarrow{\$} \mathbb{A}_\lambda$.

  By trigger scheme security, $\mathsf{G}_5 \approx \mathsf{G}_6$.

- In G_7, we clean up the components:

$$y^* = \pi(x'), \qquad s^* = \sigma(r'),$$

$$P(v\|r) = \begin{cases} \pi(x'), & \text{if } v\|\mathcal{T}_0(r) = v^*\|1; \\ \pi(v\|r), & \text{otherwise;} \end{cases} \qquad Q(r) = \begin{cases} \sigma(r'), & \text{if } \mathcal{T}_0(r) = 1; \\ \sigma(r), & \text{otherwise;} \end{cases}$$

$$R(y) = \begin{cases} \bot, & \text{if } y = y^*; \\ \pi^{-1}(y), & \text{otherwise.} \end{cases}$$

  Because the removed branches are never entered, $\mathsf{G}_6 \approx \mathsf{G}_7$ by $i\mathcal{O}$ security. Note that $\mathsf{G}_7 \equiv \mathsf{Exp}^1_{\text{cca-rr}}$ by renaming variables.

The proof concludes by hybrid argument. □

6 Unary Homomorphic Encryption

In this section, we present a homomorphic encryption scheme supporting unbounded iterations of a fixed unary function. Again, we recall the definition first.

Definition 7 (unary HE, selective security). Let

$$\mathsf{DH} = \{(D_{\lambda,\mathsf{param}}, H_{\lambda,\mathsf{param}})\}_{\lambda\in\mathbb{N},\mathsf{param}\in\mathsf{Params}_\lambda}$$

be a family of domains and non-empty function sets such that the domain of every function in $H_{\lambda,\mathsf{param}}$ is $D_{\lambda,\mathsf{param}}$. A *(unary) homomorphic encryption (HE) scheme* for DH consists of four efficient algorithms.

- $\mathsf{Gen}(1^\lambda, \mathsf{param})$ takes a functionality description param as input, and outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Enc}(1^\lambda, \mathsf{pk}, x)$ takes as input pk and some plaintext $x \in D_{\lambda,\mathsf{param}}$, and outputs a ciphertext ct.
- $\mathsf{Eval}(1^\lambda, \mathsf{pk}, h, \mathsf{ct})$ takes as input pk, some $h \in H_{\lambda,\mathsf{param}}$, and a ciphertext ct. It outputs an evaluated ciphertext ct′.
- $\mathsf{Dec}(1^\lambda, \mathsf{sk}, \mathsf{ct}')$ takes sk, ct′ as input. It outputs the decryption result.

The scheme must be *compact*, i.e., $|\mathsf{ct}'| \le \mathrm{poly}(\lambda, |\mathsf{param}|, |x|)$ does not grow with $|h|$.

The scheme must be *correct*, i.e., for all $\lambda \in \mathbb{N}$, $\mathsf{param} \in \mathsf{Params}_\lambda$, $x \in D_{\lambda,\mathsf{param}}$, $h \in H_{\lambda,\mathsf{param}}$, it holds that

$$\Pr\left[\begin{array}{ll} (\mathsf{pk},\mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda,\mathsf{param}) & \\ \quad \mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(1^\lambda,\mathsf{pk},x) & : \ \mathsf{Dec}(1^\lambda,\mathsf{sk},\mathsf{ct}') = h(x) \\ \quad \mathsf{ct}' \xleftarrow{\$} \mathsf{Eval}(1^\lambda,\mathsf{pk},h,\mathsf{ct}) & \end{array}\right] = 1.$$

The scheme is *selectively secure* if $\mathsf{Exp}^0_{\mathsf{he}} \approx \mathsf{Exp}^1_{\mathsf{he}}$, where $\mathsf{Exp}^\beta_{\mathsf{he}}(1^\lambda,\mathcal{A})$ proceeds as follows.

- **Setup/Challenge.** Launch $\mathcal{A}(1^\lambda)$ and receive from it $\mathsf{param} \in \mathsf{Params}_\lambda$ and x^*_0, $x^*_1 \in D_{\lambda,\mathsf{param}}$ of equal length. Run

  $$(\mathsf{pk},\mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda,\mathsf{param}), \qquad \mathsf{ct}^* \xleftarrow{\$} \mathsf{Enc}(1^\lambda,\mathsf{pk},x^*_\beta),$$

  and send $(\mathsf{pk},\mathsf{ct}^*)$ to $\mathcal{A}$.
- **Guess.** $\mathcal{A}$ outputs a bit β', which is the output of the experiment.

Ingredients of Construction 3. Let

- $\mathsf{PPRP} = (\mathsf{PPRP.Gen}, \mathsf{PPRP.Permute}, \mathsf{PPRP.Eval}, \mathsf{PPRP.Eval}^{-1})$ be an IP-PRP, and
- $i\mathcal{O}$ a circuit obfuscator.

Construction 3 (unary HE). Define (see Definition 7)

$$\mathsf{Params} := \left\{ g \,\middle|\, g : \{0,1\}^n \to \{0,1\}^n \text{ is a circuit for some } n \in \mathbb{N} \right\},$$
$$D_g := \{0,1\}^n, \qquad H_g := \left\{ g^{(t)} \,\middle|\, t \in \mathbb{N} \right\}, \qquad \mathsf{DH} := \{(D_g, H_g)\}_{g \in \mathsf{Params}}.$$

Here, $g^{(t)}$ is the tth iteration of g, encoded as 1^t. Our unary HE for DH works as follows.

- $\mathsf{Gen}(g)$ takes a circuit $g : \{0,1\}^n \to \{0,1\}^n$ as input. It samples PPRP instances (for domain lengths $(n+2\lambda)$ and λ)

  $$\pi \xleftarrow{\$} \mathsf{PPRP.Gen}(1^{n+2\lambda}), \qquad \sigma \xleftarrow{\$} \mathsf{PPRP.Gen}(1^\lambda).$$

  The domain of π is parsed as $\{0,1\}^n \times \{0,1\}^\lambda \times \{0,1\}^\lambda$. The algorithm defines

  $$\mathsf{Ev}(\mathsf{ct}) := \pi(g(x) \| (\ell+1) \| \sigma(r)) \qquad \text{with } x \| \ell \| r \leftarrow \pi^{-1}(\mathsf{ct}),$$

  where $(\ell+1)$ is modulo 2^λ. It runs (with appropriate paddings) and sets

  $$\widetilde{\pi} \xleftarrow{\$} i\mathcal{O}(\pi), \qquad \widetilde{\mathsf{Ev}} \xleftarrow{\$} i\mathcal{O}(\mathsf{Ev}), \qquad \mathsf{pk} \leftarrow (\widetilde{\pi}, \widetilde{\mathsf{Ev}}), \qquad \mathsf{sk} \leftarrow \pi^{-1}.$$

  The algorithm outputs $(\mathsf{pk},\mathsf{sk})$.
- $\mathsf{Enc}(\mathsf{pk},x)$ parses pk as $(\widetilde{\pi},\widetilde{\mathsf{Ev}})$, samples $\ell \xleftarrow{\$} \{0,1\}^\lambda$, $r \xleftarrow{\$} \{0,1\}^\lambda$, and computes and outputs $\mathsf{ct} \leftarrow \widetilde{\pi}(x \| \ell \| r)$.
- $\mathsf{Eval}(\mathsf{pk},1^t,\mathsf{ct})$ parses pk as $(\widetilde{\pi},\widetilde{\mathsf{Ev}})$, sets $\mathsf{ct}^{(0)} \leftarrow \mathsf{ct}$, and computes $\mathsf{ct}^{(i)} \leftarrow \widetilde{\mathsf{Ev}}(\mathsf{ct}^{(i-1)})$ for $i = 1,\ldots,t$. It outputs $\mathsf{ct}' \leftarrow \mathsf{ct}^{(t)}$. (Here, $h = g^{(t)}$.)

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}')$ parses sk as π^{-1}. It computes $x' \| \ell' \| r' \leftarrow \pi^{-1}(\mathsf{ct}')$ and outputs x'.

Correctness can be verified easily from that of the ingredients. Compactness holds, as $|\mathsf{ct}'| = |x| + 2\lambda \le \mathrm{poly}(\lambda, |\mathsf{param}|, |x|)$.

Theorem 8 (¶). *Suppose $i\mathcal{O}$, PPRP are subexponentially secure (Definitions 1 and 2) and PPRP supports all transpositions (see Lemma 1), then Construction 3 is selectively secure (Definition 7).*

The analysis of reduction time and advantage is in the proof.

6.1 Lemma(s)

To prove the security of Construction 3, we change a prefix of the challenge evaluation chain (starting from ct^*) to pseudorandom, up to a random subexponential length. We discuss two lemmas that will be handy for this process.

Recall that Lemma 7 says that a ciphertext $y^* \leftarrow \pi(v^* \| r^*)$ remains pseudorandom in the presence of a punctured decryption program $i\mathcal{O}(\pi^{-1} \setminus \{y^*\})$, even when an independent PPRP σ is used to randomize[13] the encryption randomness r^* into $s^* \leftarrow \sigma(r^*)$, and that s^* is jointly pseudorandom. In Construction 3, the program $\widetilde{\mathsf{Ev}}$ in pk can decrypt any ciphertext internally, so although we are only proving CPA security, the underlying PKE must remain secure given a punctured decryption program.

Lemma 9 says that a *permutable* PRP over a large enough domain[14] is *puncturable* (as in puncturable PRF) at any fixed input, with the additional benefit of having access to a punctured inverse. It is useful for efficiently *recognizing* (and generating) a set of (distinct) pseudorandom values, which will be the replaced prefix in the proof of Theorem 8. We further discuss puncturable PRP in Section 7.3, including reconciliation with [BKW17].

We restate Lemma 7 (in a simplified form) below, then state and prove Lemma 9.

Lemma 7 (restated, simplified). *For any adversarially chosen $v^* \in \{0,1\}^\star$,*

$$\begin{aligned} &\left\{\left(\pi(v^* \| r^*),\ \sigma(r^*),\ i\mathcal{O}(\pi),\ i\mathcal{O}(\sigma),\ i\mathcal{O}(\pi^{-1} \setminus \{\pi(v^* \| r^*)\})\right)\right\} \\ \approx &\left\{\left(\quad y_1 \quad,\quad s_1 \quad,\ i\mathcal{O}(\pi),\ i\mathcal{O}(\sigma),\ i\mathcal{O}(\pi^{-1} \setminus \{\quad y_1 \quad\})\right)\right\}, \end{aligned}$$

where r^* (λ-bit), y_1, s_1 are uniform and π, σ are PRP input-permutable by transpositions. The advantage relation is $\varepsilon_{\text{cca-rr}} = 11\varepsilon_{i\mathcal{O}} + 3\varepsilon_{\text{trigger}} + 2\varepsilon_{\mathsf{PPRP}}$.

Lemma 9 (large-domain PPRP can be punctured; ¶). *Let $i\mathcal{O}$ be an indistinguishability obfuscator and $\mathsf{PPRP} = (\mathsf{Gen}, \mathsf{Permute}, \mathsf{Eval}, \mathsf{Eval}^{-1})$ an IP-PRP supporting all transpositions (Definitions 1 and 2 and Lemma 1), then $\mathsf{Exp}^0_{\text{pnctr}} \approx \mathsf{Exp}^1_{\text{pnctr}}$, where $\mathsf{Exp}^\beta_{\text{pnctr}}(\mathcal{A})$ proceeds as follows.*

[13] The scheme is not a "rerandomizable encryption" in the usual sense, which is about generating statistically fresh samples of ciphertexts of the same plaintext and whose rerandomization procedure is randomized given $(\mathsf{pk}, \mathsf{ct})$. Here, the refreshed randomness is determined by $(\mathsf{pk}, \mathsf{ct})$ and it is only used for a single ciphertext, namely the next homomorphically evaluated one.

[14] The lemma applies to domains of size $2^{\lambda^{\Omega(1)}}$ after tuning the security parameter. Smaller sizes can be allowed by assuming, e.g., subexponentially secure OWF. However, to puncture PRP, the domain size cannot be a fixed $\mathrm{poly}(\lambda)$, unlike in PPRF, as the image at the punctured point is efficiently determined by the images at the other points.

- **Challenge.** Launch $\mathcal{A}()$ and receive $v^* \in \{0,1\}^\star$ from it. Run

$$\varphi \xleftarrow{\$} \mathsf{Gen}(1^{|v^*|}), \qquad y_0 \leftarrow \varphi(v^*), \qquad y_1 \xleftarrow{\$} \{0,1\}^{|v^*|},$$
$$y^* \leftarrow y_\beta, \qquad \widetilde{P} \xleftarrow{\$} i\mathcal{O}(\varphi \setminus \{v^*\}), \qquad \widetilde{Q} \xleftarrow{\$} i\mathcal{O}(\varphi^{-1} \setminus \{y^*\}),$$

  where the calls to $i\mathcal{O}$ are appropriately padded. Send $(y^*, \widetilde{P}, \widetilde{Q})$ to $\mathcal{A}$.

- **Guess.** $\mathcal{A}$ outputs a bit β'. The output of the experiment is β' if $|v^*| \geq \lambda$; otherwise, the output is set to 0.

The proof uses time-preserving reductions and $\varepsilon_{\text{pnctr}} = 6\varepsilon_{i\mathcal{O}} + \varepsilon_{\text{trigger}} + \varepsilon_{\text{PPRP}}$.

Proof (Lemma 9). Sample $v' \xleftarrow{\$} \{0,1\}^{|v^*|}$. Let $(\mathbb{A}, \Delta)$ be a trigger scheme (potentially subexponentially secure) promised by Lemma 2 and $\mathcal{T}$ a trigger potentially activated at v'. (We assume $|v^*| \geq \lambda$ as otherwise, the output is always 0.) Writing P, Q for the programs being obfuscated into $\widetilde{P}$, $\widetilde{Q}$, we consider the following hybrids, which are fully specified by y^*, P, Q.

- G_0 is $\mathsf{Exp}^0_{\text{pnctr}}$, where

$$y^* = \varphi(v^*),$$
$$P(v) = \begin{cases} \bot, & \text{if } v = v^*; \\ \varphi(v), & \text{otherwise;} \end{cases} \qquad Q(y) = \begin{cases} \bot, & \text{if } y = y^*; \\ \varphi^{-1}(y), & \text{otherwise.} \end{cases}$$

- In G_1, we sample $\mathcal{T} \xleftarrow{\$} \mathbb{A}$ and embed it into the programs:

$$y^* = \varphi(v^*),$$
$$P(v) = \begin{cases} \bot, & \text{if } v = v^*; \\ \boxed{\varphi(v^*),} & \boxed{\text{else if } \mathcal{T}(v) = 1;} \\ \varphi(v), & \text{otherwise;} \end{cases} \quad Q(y) = \begin{cases} \bot, & \text{if } y = y^*; \\ \boxed{v^*,} & \boxed{\text{else if } \mathcal{T}(\varphi^{-1}(y)) = 1;} \\ \varphi^{-1}(y), & \text{otherwise.} \end{cases}$$

  Since the newly added branches are never entered, $\mathsf{G}_0 \approx \mathsf{G}_1$ by $i\mathcal{O}$ security.

- In G_2, we switch $\mathcal{T}$ to $\mathcal{T} \xleftarrow{\$} \boxed{\Delta(v')}$.

  By trigger scheme security, we have $\mathsf{G}_1 \approx \mathsf{G}_2$.

- In G_3, we run $\varphi_* \xleftarrow{\$} \mathsf{Permute}(\varphi, (v^* \;\; v'), 0)$ and rewrite everything to use φ_*, v':

$$y^* = \boxed{\varphi_*}(v^*),$$
$$P(v) = \begin{cases} \bot, & \text{if } v = v^*; \\ \boxed{\varphi_*}(v^*), & \text{else if } \boxed{v = v'}; \\ \boxed{\varphi_*}(v), & \text{otherwise;} \end{cases} \quad Q(y) = \begin{cases} \bot, & \text{if } \boxed{\varphi_*^{-1}(y) = v^*}; \\ v^*, & \text{else if } \boxed{\varphi_*^{-1}(y) = v'}; \\ \boxed{\varphi_*^{-1}}(y), & \text{otherwise.} \end{cases}$$

  By the correctness of PPRP and the trigger scheme, y^* and the functionalities of P, Q are unchanged. Therefore, $\mathsf{G}_2 \approx \mathsf{G}_3$ by $i\mathcal{O}$ security.

- In G_4, we permute the key by $\varphi_* \xleftarrow{\$} \mathsf{Permute}(\varphi, (v^* \;\; v'), \boxed{1})$.

  $\mathsf{G}_3 \approx \mathsf{G}_4$ by PPRP security.

- In G_5, we clean up the components:

$$y^* = \boxed{\varphi(v')},$$

$$\boxed{P}(v) = \begin{cases} \bot, & \text{if } v = v^*; \\ \varphi(v), & \text{otherwise;} \end{cases} \qquad \boxed{Q}(y) = \begin{cases} \bot, & \text{if } y = y^*; \\ \varphi^{-1}(y), & \text{otherwise.} \end{cases}$$

By PPRP correctness, y^* and the functionalities of P, Q are unchanged, so we have $\mathsf{G}_4 \approx \mathsf{G}_5$ by $i\mathcal{O}$ security. Note that $\mathsf{G}_5 \equiv \mathsf{Exp}^1_{\mathrm{pnctr}}$ by renaming $\varphi(v')$ to y_1.

The proof concludes by hybrid argument. □

6.2 Security Proof

Proof (Theorem 8). Let $(\mathcal{A}, \Delta)$ be a *subexponentially secure* trigger scheme promised by Lemma 2. Let $0 < c < 1$ be a constant to be determined later and define $L := 2^{\lfloor \lambda^c \rfloor}$.

Sample

$$\ell^* \xleftarrow{\$} \{0,1\}^\lambda, \qquad r^* \xleftarrow{\$} \{0,1\}^\lambda, \qquad \widehat{\ell^*} \xleftarrow{\$} [0, L), \qquad \varphi \xleftarrow{\$} \mathsf{PPRP.Gen}(1^{n+2\lambda}).$$

Here, ℓ^* and r^* are used for ct^* in $\mathsf{Exp}^\beta_{\mathrm{he}}$ (but not necessarily so in the other hybrids), $\widehat{\ell^*}$ is written in binary and exactly $\lfloor \lambda^c \rfloor$ bits, and the domain/codomain of φ are conveniently regarded as $[0, 2^{n+2\lambda})$ by binary representation of natural numbers. We generate π, σ as in the scheme and use $\mathsf{pk} \xleftarrow{\$} (i\mathcal{O}(\pi), i\mathcal{O}(\mathsf{Ev}))$. Denote by $\mathcal{T}$ a trigger potentially activated at $\widehat{\ell^*}$.

We first plant a trigger to stop evaluation whenever the input ct to Ev has $\ell = \ell^* + \widehat{\ell^*}$ (compared modulo 2^λ). This puts an upper bound, namely L, on the length of the chain of ciphertexts reachable from ct^* ("the challenge chain"). After that, we replace a longer and longer prefix of the challenge chain by pseudorandom, efficiently generated and recognized by PPRP φ. (The technique of controlling chain length by trigger is from [BPW16].)

Formally, consider the following hybrids, which are fully specified by ct^*, Ev.

- H^β_0 is $\mathsf{Exp}^\beta_{\mathrm{he}}$, i.e.,

$$\mathsf{ct}^* = \pi(x^*_\beta \| \ell^* \| r^*), \quad \mathsf{Ev}(\mathsf{ct}) = \pi(g(x) \| (\ell + 1) \| \sigma(r)) \quad \text{with } x \| \ell \| r \leftarrow \pi^{-1}(\mathsf{ct}).$$

- In H^β_1, we add a deactivated trigger $\mathcal{T} \xleftarrow{\$} \mathcal{A}(1^{\lfloor \lambda^c \rfloor}, 1^{\lfloor \lambda^c \rfloor})$[15] to Ev:

$$\mathsf{ct}^* = \pi(x^*_\beta \| \ell^* \| r^*),$$

$$\mathsf{Ev}(\mathsf{ct}) = \begin{cases} \boxed{\bot,} & \boxed{\begin{array}{l}\text{if } \pi^{-1}(\mathsf{ct}) = x \| \ell \| r \text{ with} \\ \quad \ell - \ell^* = 0^{\lambda - \lfloor \lambda^c \rfloor} \| \widehat{\ell} \text{ and } \mathcal{T}(\widehat{\ell}) = 1;\end{array}} \\ \pi(g(x) \| (\ell + 1) \| \sigma(r)), & \text{otherwise, } \pi^{-1}(\mathsf{ct}) = x \| \ell \| r. \end{cases}$$

Here, $(\ell - \ell^*)$ is modulo 2^λ.

Since the newly added branch is never entered, $\mathsf{H}^\beta_0 \approx \mathsf{H}^\beta_1$ by $i\mathcal{O}$ security.

[15] To adhere to the formalism of Lemma 2 (for security, trigger point length cannot be less than trigger scheme security parameter), the security parameter for the trigger scheme is tuned down. The point is to use a smaller-range trigger, not a less secure trigger. However, complexity "deleveraging" complicates uniform reductions.

- In H_2^β, we activate the trigger by $\mathcal{T} \xleftarrow{\$} \boxed{\Delta(1^{\lfloor \lambda^c \rfloor}, \widehat{\ell^*})}$. Here, $\widehat{\ell^*}$ is written in binary and exactly $\lfloor \lambda^c \rfloor$ bits.

  By trigger scheme security, we have $\mathsf{H}_1^\beta \approx \mathsf{H}_2^\beta$.

- In H_3^β, we rewrite Ev to use $\widehat{\ell^*}$ directly and also format it for later hybrids:

$$\mathsf{ct}^* = \pi(x_\beta^* \| \ell^* \| r^*),$$

$$\mathsf{Ev}(\mathsf{ct}) = \begin{cases} \boxed{\varphi(\varphi^{-1}(\mathsf{ct}) + 1),} & \boxed{\text{if } 0 \le \varphi^{-1}(\mathsf{ct}) < -1;} \\ \boxed{\mathsf{ct}^*,} & \boxed{\text{else if } \varphi^{-1}(\mathsf{ct}) = -1;} \\ \bot, & \boxed{\text{else}} \text{ if } \pi^{-1}(\mathsf{ct}) = x \| \ell \| r \text{ with } \boxed{\ell = \ell^* + \widehat{\ell^*}}; \\ \pi(g(x) \| (\ell + 1) \| \sigma(r)), & \text{otherwise, } \pi^{-1}(\mathsf{ct}) = x \| \ell \| r. \end{cases}$$

  Here, $\ell \stackrel{?}{=} \ell^* + \widehat{\ell^*}$ is compared modulo 2^λ, yet $\varphi^{-1}(\mathsf{ct})$ is not modulo anything. Note that $\varphi^{-1}(\mathsf{ct})$ is always interpreted as a non-negative integer, so the newly introduced branches are never hit.

  By $i\mathcal{O}$ security, $\mathsf{H}_2^\beta \approx \mathsf{H}_3^\beta$.

- In $\mathsf{H}_{4,\mathsf{t}}^\beta$ (for $\mathsf{t} \in [-1, L)$), a prefix of the challenge chain is replaced. Let

$$\mathsf{t}^* = \min\{\mathsf{t}, \widehat{\ell^*}\}, \qquad \mathsf{ct}_\beta^{(\mathsf{t}^*+1)} = \begin{cases} \bot, & \text{if } \mathsf{t}^* = \widehat{\ell^*}; \\ \pi(g^{(\mathsf{t}^*+1)}(x_\beta^*) \| (\ell^* + \mathsf{t}^* + 1) \| r^*), & \text{otherwise.} \end{cases}$$

  The components are

$$\mathsf{ct}^* = \begin{cases} \pi(x_\beta^* \| \ell^* \| r^*), & \text{if } \mathsf{t}^* = -1; \\ \varphi(0), & \text{otherwise;} \end{cases}$$

$$\mathsf{Ev}(\mathsf{ct}) = \begin{cases} \varphi(\varphi^{-1}(\mathsf{ct}) + 1), & \text{if } 0 \le \varphi^{-1}(\mathsf{ct}) < \boxed{\mathsf{t}^*}; \\ \boxed{\mathsf{ct}_\beta^{(\mathsf{t}^*+1)}}, & \text{else if } \varphi^{-1}(\mathsf{ct}) = \boxed{\mathsf{t}^*}; \\ \bot, & \text{else if } \pi^{-1}(\mathsf{ct}) = x \| \ell \| r \text{ with } \ell = \ell^* + \widehat{\ell^*}; \\ \pi(g(x) \| (\ell + 1) \| \sigma(r)), & \text{otherwise, } \pi^{-1}(\mathsf{ct}) = x \| \ell \| r. \end{cases}$$

  Here, $\mathsf{ct}_\beta^{(\mathsf{t}^*+1)}$ is written in Ev as a value, either $\bot$ or an $(n + 2\lambda)$-bit string, not as a computation iterating g.

  Observe that $\mathsf{H}_3^\beta \equiv \mathsf{H}_{4,-1}^\beta$.

  Claim 10 (¶). $\mathsf{H}_{4,\mathsf{t}-1}^\beta \approx \mathsf{H}_{4,\mathsf{t}}^\beta$ *for each* $\mathsf{t} \in [0, L)$ *and* $\beta \in \{0, 1\}$.

  In $\mathsf{H}_{4,L-1}^\beta$, we have $\mathsf{ct}^* = \varphi(0)$ and $\mathsf{ct}_\beta^{(\mathsf{t}^*+1)} = \bot$, so β does not appear in ct^* or (in particular, the third branch of) Ev, hence the adversary's view. It follows that $\mathsf{H}_{4,L-1}^0 \equiv \mathsf{H}_{4,L-1}^1$.

It remains to determine c and analyze the security loss. We have

$$\text{(uniform)} \quad |\varepsilon_{\mathrm{he}}(\lambda)| \le 4|\varepsilon_{i\mathcal{O}}(\lambda)| + 2\sqrt{(\lfloor \lambda^c \rfloor + 1)^{-1+1/c}/c \cdot \varepsilon_{\mathrm{trigger},1}(\lfloor \lambda^c \rfloor)} + 2L \cdot (20|\varepsilon_{i\mathcal{O}}(\lambda)| + 4|\varepsilon_{\mathrm{trigger},2}(\lambda)| + 3|\varepsilon_{\mathsf{PPRP}}(\lambda)| + L \cdot 2^{-2\lambda}),$$

(non-uniform) $|\varepsilon_{\mathrm{he}}(\lambda)| \le 4|\varepsilon_{i\mathcal{O}}(\lambda)| + 2|\varepsilon_{\mathrm{trigger},1}(\lfloor\lambda^c\rfloor)|$
$+ 2L \cdot (20|\varepsilon_{i\mathcal{O}}(\lambda)| + 4|\varepsilon_{\mathrm{trigger},2}(\lambda)| + 3|\varepsilon_{\mathsf{PPRP}}(\lambda)| + L \cdot 2^{-2\lambda})$.

Here, $\varepsilon_{\mathrm{trigger},1}$ and $\varepsilon_{\mathrm{trigger},2}$ indicate that the losses arise from two different reductions. Let $0 < c_{i\mathcal{O}}, c_{\mathrm{trigger}}, c_{\mathsf{PPRP}} < 1$ be the constants for the subexponential security of $i\mathcal{O}$, trigger, PPRP. It suffices to pick $0 < c < \min\{c_{i\mathcal{O}}, c_{\mathrm{trigger}}, c_{\mathsf{PPRP}}\}$ to obtain $\varepsilon_{\mathrm{he}} = \mathrm{negl}(\lambda)$ as desired.

If the reductions must be uniform, they could take additional time up to $\widetilde{\Theta}(L)$ for computing $g^{(\mathsf{t}^*+1)}(x^*_\beta)$ for one $\mathsf{t}^* \in [-1, L)$. Moreover, c must be picked to make $\lambda \mapsto \lfloor\lambda^c\rfloor$ easily computable. Alternatively, they could receive $\lfloor\lambda^c\rfloor$ and the needed $g^{(\mathsf{t}^*+1)}(x^*_\beta)$ as advice, which shortens their time to roughly that of the (underlying) adversary. □

Proof (Claim 10). We rely on Lemmas 7 and 9. When $\mathsf{t} > \widehat{\ell}^*$ (an event that happens with some probability), the two hybrids are identical, so the reductions can simply abort. Let's focus on the interesting case of $\mathsf{t} \le \widehat{\ell}^*$. This provides $\min\{\mathsf{t}-1, \widehat{\ell}^*\} = \mathsf{t}-1$, $\min\{\mathsf{t}, \widehat{\ell}^*\} = \mathsf{t}$, and we will use $(\mathsf{t}-1)$, t instead of t^* etc. Write $x' = g^{(\mathsf{t})}(x^*_\beta)$. Consider the following hybrids.

- G_0 is $\mathsf{H}^\beta_{4,\mathsf{t}-1}$. We reformat the components for the current proof:

$$y^* = \pi(x' \| (\ell^* + \mathsf{t}) \| r^*),$$

$$z^* = \begin{cases} \bot, & \text{if } \mathsf{t} = \widehat{\ell}^*; \\ \pi(g(x') \| (\ell^* + \mathsf{t} + 1) \| s^*), & \text{otherwise, with } s^* \leftarrow \sigma(r^*); \end{cases}$$

$$\mathsf{ct}^* = \begin{cases} y^*, & \text{if } \mathsf{t} = 0; \\ \varphi(0), & \text{otherwise;} \end{cases}$$

$$\mathsf{Ev}(\mathsf{ct}) = \begin{cases} \varphi(\varphi^{-1}(\mathsf{ct}) + 1), & \text{if } 0 \le \varphi^{-1}(\mathsf{ct}) < \mathsf{t}-1; \\ y^*, & \text{else if } \varphi^{-1}(\mathsf{ct}) = \mathsf{t}-1; \\ \bot, & \text{else if } \pi^{-1}(\mathsf{ct}) = x \| \ell \| r \text{ with } \ell = \ell^* + \widehat{\ell}^*; \\ \pi(g(x) \| (\ell+1) \| \sigma(r)), & \text{otherwise, } \pi^{-1}(\mathsf{ct}) = x \| \ell \| r. \end{cases}$$

- In G_1, we prepare for Lemma 7 as follows:

$$y^* = \pi(x' \| (\ell^* + \mathsf{t}) \| r^*), \quad \boxed{\widetilde{P} \xleftarrow{\$} i\mathcal{O}(\pi), \quad \widetilde{Q} \xleftarrow{\$} i\mathcal{O}(\sigma), \quad \widetilde{R} \xleftarrow{\$} i\mathcal{O}(\pi^{-1} \setminus \{y^*\})},$$

$$z^* = \begin{cases} \bot, & \text{if } \mathsf{t} = \widehat{\ell}^*; \\ \boxed{\widetilde{P}}(g(x') \| (\ell^* + \mathsf{t} + 1) \| s^*), & \text{otherwise, with } s^* \leftarrow \sigma(r^*); \end{cases}$$

$$\mathsf{ct}^* = \begin{cases} y^*, & \text{if } \mathsf{t} = 0; \\ \varphi(0), & \text{otherwise;} \end{cases}$$

$$\mathsf{Ev}(\mathsf{ct}) = \begin{cases} \varphi(\varphi^{-1}(\mathsf{ct}) + 1), & \text{if } 0 \le \varphi^{-1}(\mathsf{ct}) < \mathsf{t}-1; \\ y^*, & \text{else if } \varphi^{-1}(\mathsf{ct}) = \mathsf{t}-1; \\ \boxed{z^*}, & \boxed{\text{else if } \mathsf{ct} = y^*;} \\ \bot, & \text{else if } \boxed{\widetilde{R}}(\mathsf{ct}) = x \| \ell \| r \text{ with } \ell = \ell^* + \widehat{\ell}^*; \\ \boxed{\widetilde{P}}(g(x) \| (\ell+1) \| \boxed{\widetilde{Q}}(r)), & \text{otherwise, } \boxed{\widetilde{R}}(\mathsf{ct}) = x \| \ell \| r. \end{cases}$$

The values of y^*, z^*, ct^* are unchanged. Note that Ev has the same functionality in G_0, G_1: if y^* is not handled by the first two branches, then z^* is indeed its output produced by the last two branches; the program $\widetilde{R}$ is equivalent to π^{-1} except at y^*, which is never fed into $\widetilde{R}$ thanks to the newly introduced branch.

We have $\mathsf{G}_0 \approx \mathsf{G}_1$ by $i\mathcal{O}$ security.

- In G_2, we change how y^*, z^* are computed:

$$\boxed{y^* \xleftarrow{\$} \{0,1\}^{n+2\lambda}}, \quad \widetilde{P} \xleftarrow{\$} i\mathcal{O}(\pi), \quad \widetilde{Q} \xleftarrow{\$} i\mathcal{O}(\sigma), \quad \widetilde{R} \xleftarrow{\$} i\mathcal{O}(\pi^{-1} \setminus \{y^*\}),$$

$$z^* = \begin{cases} \bot, & \text{if } \mathsf{t} = \widehat{\ell^*}; \\ \widetilde{P}(g(x') \| (\ell^* + \mathsf{t} + 1) \| s^*), & \text{otherwise, with } \boxed{s^* \xleftarrow{\$} \{0,1\}^\lambda}; \end{cases}$$

$$\mathsf{ct}^* = \begin{cases} y^*, & \text{if } \mathsf{t} = 0; \\ \varphi(0), & \text{otherwise;} \end{cases}$$

$$\mathsf{Ev}(\mathsf{ct}) = \begin{cases} \varphi(\varphi^{-1}(\mathsf{ct}) + 1), & \text{if } 0 \le \varphi^{-1}(\mathsf{ct}) < \mathsf{t} - 1; \\ y^*, & \text{else if } \varphi^{-1}(\mathsf{ct}) = \mathsf{t} - 1; \\ z^*, & \text{else if } \mathsf{ct} = y^*; \\ \bot, & \text{else if } \widetilde{R}(\mathsf{ct}) = x \| \ell \| r \text{ with } \ell = \ell^* + \widehat{\ell^*}; \\ \widetilde{P}(g(x) \| (\ell + 1) \| \widetilde{Q}(r)), & \text{otherwise, } \widetilde{R}(\mathsf{ct}) = x \| \ell \| r. \end{cases}$$

$\mathsf{G}_1 \approx \mathsf{G}_2$ reduces to Lemma 7. (The reduction chooses $v^* = x' \| (\ell^* + \mathsf{t})$.)

- In G_3, we remove obfuscations of π, σ, π^{-1} and add obfuscations of φ, φ^{-1} to prepare for Lemma 9. The components become

$$y^* \xleftarrow{\$} \{0,1\}^{n+2\lambda}, \quad \widetilde{P} \xleftarrow{\$} i\mathcal{O}(\pi), \quad \widetilde{Q} \xleftarrow{\$} i\mathcal{O}(\sigma), \quad \widetilde{R} \xleftarrow{\$} i\mathcal{O}(\pi^{-1} \setminus \{y^*\}),$$

$$\boxed{\widetilde{P'} \xleftarrow{\$} i\mathcal{O}(\varphi \setminus \{\mathsf{t}\}), \quad \widetilde{Q'} \xleftarrow{\$} i\mathcal{O}(\varphi^{-1} \setminus \{y^*\}),}$$

$$z^* = \begin{cases} \bot, & \text{if } \mathsf{t} = \widehat{\ell^*}; \\ \boxed{\pi}(g(x') \| (\ell^* + \mathsf{t} + 1) \| s^*), & \text{otherwise, with } s^* \xleftarrow{\$} \{0,1\}^\lambda; \end{cases}$$

$$\mathsf{ct}^* = \begin{cases} y^*, & \text{if } \mathsf{t} = 0; \\ \boxed{\widetilde{P'}}(0), & \text{otherwise;} \end{cases}$$

$$\mathsf{Ev}(\mathsf{ct}) = \begin{cases} \boxed{\widetilde{P'}}(\boxed{\widetilde{Q'}}(\mathsf{ct}) + 1), & \text{if } 0 \le \boxed{\widetilde{Q'}}(\mathsf{ct}) < \mathsf{t} - 1; \\ y^*, & \text{else if } \boxed{\widetilde{Q'}}(\mathsf{ct}) = \mathsf{t} - 1; \\ z^*, & \text{else if } \mathsf{ct} = y^*; \\ \bot, & \text{else if } \boxed{\pi^{-1}}(\mathsf{ct}) = x \| \ell \| r \text{ with } \ell = \ell^* + \widehat{\ell^*}; \\ \boxed{\pi}(g(x) \| (\ell + 1) \| \boxed{\sigma}(r)), & \text{otherwise, } \boxed{\pi^{-1}}(\mathsf{ct}) = x \| \ell \| r. \end{cases}$$

The values of y^*, z^* are unchanged. As for ct^*: when $\mathsf{t} = 0$, the branch $\mathsf{ct}^* = y^*$ is taken (the puncturing of $\widetilde{P'}$ is irrelevant); when $\mathsf{t} \neq 0$, we have $\mathsf{ct}^* = \widetilde{P'}(0) = \varphi(0)$ as $\widetilde{P'}$ is not punctured at 0. Therefore, ct^* is also unchanged.

The functionality of Ev is changed only when $\varphi^{-1}(y^*) \in [0, \mathsf{t})$ ("bad event"). Since y^* is uniformly random over $(n + 2\lambda)$-bit strings conditioned on n, t, we have

$$\Pr[\text{ bad event} \mid n, \mathsf{t}\,] \le \mathsf{t} \cdot 2^{-n-2\lambda} \le L \cdot 2^{-2\lambda},$$

and by the law of total probability, $\Pr[\text{bad event}] \le L \cdot 2^{-2\lambda}$.

By $i\mathcal{O}$ security, $\mathsf{G}_2 \approx \mathsf{G}_3$.

- In H_4, we "collect" y^* into φ:

$$\boxed{y^* \leftarrow \varphi(\mathsf{t})}, \quad \widetilde{P}' \xleftarrow{\$} i\mathcal{O}(\varphi \setminus \{\mathsf{t}\}), \quad \widetilde{Q}' \xleftarrow{\$} i\mathcal{O}(\varphi^{-1} \setminus \{y^*\}),$$

$$z^* = \begin{cases} \bot, & \text{if } \mathsf{t} = \widehat{\ell}^*; \\ \pi(g(x') \| (\ell^* + \mathsf{t} + 1) \| s^*), & \text{otherwise, with } s^* \xleftarrow{\$} \{0,1\}^\lambda; \end{cases}$$

$$\mathsf{ct}^* = \begin{cases} y^*, & \text{if } \mathsf{t} = 0; \\ \widetilde{P}'(0), & \text{otherwise;} \end{cases}$$

$$\mathsf{Ev}(\mathsf{ct}) = \begin{cases} \widetilde{P}'(\widetilde{Q}'(\mathsf{ct}) + 1), & \text{if } \ 0 \le \widetilde{Q}'(\mathsf{ct}) < \mathsf{t} - 1; \\ y^*, & \text{else if } \widetilde{Q}'(\mathsf{ct}) = \mathsf{t} - 1; \\ z^*, & \text{else if } \mathsf{ct} = y^*; \\ \bot, & \text{else if } \pi^{-1}(\mathsf{ct}) = x \| \ell \| r \text{ with } \ell = \ell^* + \widehat{\ell}^*; \\ \pi(g(x) \| (\ell + 1) \| \sigma(r)), & \text{otherwise, } \pi^{-1}(\mathsf{ct}) = x \| \ell \| r. \end{cases}$$

  $\mathsf{G}_3 \approx \mathsf{G}_4$ reduces to Lemma 9. (The transition is "in reverse direction".)

- In G_5, we undo obfuscations of φ, φ^{-1} and clean up ct^*, Ev:

$$\cancel{y^* \leftarrow \varphi(\mathsf{t}), \quad \widetilde{P}' \xleftarrow{\$} i\mathcal{O}(\varphi \setminus \{\mathsf{t}\}), \quad \widetilde{Q}' \xleftarrow{\$} i\mathcal{O}(\varphi^{-1} \setminus \{y^*\}),}$$

$$z^* = \begin{cases} \bot, & \text{if } \mathsf{t} = \widehat{\ell}^*; \\ \pi(g(x') \| (\ell^* + \mathsf{t} + 1) \| s^*), & \text{otherwise, with } s^* \xleftarrow{\$} \{0,1\}^\lambda; \end{cases}$$

$$\mathsf{ct}^* = \boxed{\varphi(0)},$$

$$\boxed{\mathsf{Ev}}(\mathsf{ct}) = \begin{cases} \varphi(\varphi^{-1}(\mathsf{ct}) + 1), & \text{if } \ 0 \le \varphi^{-1}(\mathsf{ct}) < \mathsf{t}; \\ z^*, & \text{else if } \varphi^{-1}(\mathsf{ct}) = \mathsf{t}; \\ \bot, & \text{else if } \pi^{-1}(\mathsf{ct}) = x \| \ell \| r \text{ with } \ell = \ell^* + \widehat{\ell}^*; \\ \pi(g(x) \| (\ell + 1) \| \sigma(r)), & \text{otherwise, } \pi^{-1}(\mathsf{ct}) = x \| \ell \| r. \end{cases}$$

  The value of z^* is unchanged. In G_4, for ct^*: when $\mathsf{t} = 0$, we have $\mathsf{ct}^* = y^* = \varphi(0)$; when $\mathsf{t} \ne 0$, we have $\mathsf{ct}^* = \widetilde{P}'(0) = \varphi(0)$. Therefore, ct^* is unchanged either way. The functionality of Ev is preserved — branches 1/2/3/4/5 in G_4 correspond to 1/1/2/3/4 in G_5, respectively.

  We have $\mathsf{G}_4 \approx \mathsf{G}_5$ by $i\mathcal{O}$ security. Observe that $\mathsf{G}_5 \equiv \mathsf{H}^{\beta}_{4,\mathsf{t}}$ by renaming variables.

By expanding (inlining) Lemmas 7 and 9, the overall advantage relation for Claim 10 is $|\varepsilon_{\text{hyb}}| \le 20|\varepsilon_{i\mathcal{O}}| + 4|\varepsilon_{\text{trigger}}| + 3|\varepsilon_{\text{PPRP}}| + L \cdot 2^{-2\lambda}$. □

6.3 Barrier to Richer Homomorphisms (Collision Resistance)

Aside from being unary, which is uncommon for homomorphic encryption schemes, Construction 3 only supports iterations of a *single* function. We show that unary HE supporting unbounded interleaved iterations of *two* functions implies somewhere extractable hashing, which in turn implies collision resistance. Combined with the

black-box barrier due to [AS15], two-function unary HE is difficult to construct from only (subexponentially secure) $i\mathcal{O}$ and one-way functions using known techniques.

Definition 8 (somewhere extractable hash functions). A *somewhere extractable hash (SEH) family* consists of three efficient algorithms.

- $\mathsf{HashGen}(1^\lambda, N, i^*)$ takes as input the input length N and the extraction index $i^* \in [N]$. It outputs a pair $(\mathsf{hk}, \mathsf{td})$ of hash key and trapdoor.
- $\mathsf{Hash}(1^\lambda, \mathsf{hk}, x)$ takes as input hk and $x \in \{0,1\}^N$. It is deterministic and its output h is shorter than x (when N is at least some polynomial in λ).
- $\mathsf{Extract}(1^\lambda, \mathsf{td}, h)$ takes td, h as input. It outputs a bit.

The hash family must be *correct,* i.e., for all $\lambda \in \mathbb{N}$, $N \in \mathbb{N}_+$, $i^* \in [N]$, $x \in \{0,1\}^N$, it holds that

$$\Pr\left[(\mathsf{hk}, \mathsf{td}) \xleftarrow{\$} \mathsf{HashGen}(1^\lambda, 1^N, i^*) \ : \ \mathsf{Extract}(1^\lambda, \mathsf{td}, \mathsf{Hash}(1^\lambda, \mathsf{hk}, x)) = x_{i^*} \right] = 1.$$

It is *index-hiding* if $\mathsf{Exp}^0_{\mathsf{seh}} \approx \mathsf{Exp}^1_{\mathsf{seh}}$, where $\mathsf{Exp}^\beta_{\mathsf{seh}}(1^\lambda, \mathcal{A})$ proceeds as follows.

- **Challenge.** Launch $\mathcal{A}(1^\lambda)$ and receive from it $N \in \mathbb{N}_+$ and $i^*_0, i^*_1 \in [N]$. Run

  $$(\mathsf{hk}, \mathsf{td}) \xleftarrow{\$} \mathsf{HashGen}(1^\lambda, N, i^*_\beta)$$

  and send hk to $\mathcal{A}$.
- **Guess.** $\mathcal{A}$ outputs a bit β', which is the output of the experiment.

Although Definition 8 does not come with local opening/verification, it still implies somewhere statistical binding and collision resistance. Furthermore, Construction 4 below is eligible for the generic transformation in [HJKS22; Section A] to achieve local opening/verification.

Ingredient of Construction 4. Let $\mathsf{HE} = (\mathsf{HE.Gen}, \mathsf{HE.Enc}, \mathsf{HE.Eval}, \mathsf{HE.Dec})$ be a unary HE scheme for the following DH (see Definition 7). Each param is a tuple $(g_0, g_1, \overline{T})$ consisting of two circuits $g_0, g_1 : \{0,1\}^n \to \{0,1\}^n$ and a maximum number of iterations $\overline{T} \in \mathbb{N}_+$. It specifies $D_{g_0,g_1,\overline{T}} = \{0,1\}^n$ and

$$H_{g_0,g_1,\overline{T}} = \left\{ g_{b_t} \circ \cdots \circ g_{b_1} \,\middle|\, t \in [0, \overline{T}],\ b_1, \ldots, b_t \in \{0,1\} \right\}.$$

Here, $\overline{T}$ is written in binary and $g_{b_t} \circ \cdots \circ g_{b_1}$ is represented by $(b_1, \ldots, b_t) \in \{0,1\}^t$.

Construction 4 (SEH from two-function unary HE). It works as follows.

- $\mathsf{HashGen}(N, i^*)$ defines

  $$g_b(i^*, i, v) = \begin{cases} (i^*,\ i+1,\ b), & \text{if } i+1 = i^*; \\ (i^*,\ i+1,\ v), & \text{if } i+1 \neq i^*; \end{cases} \qquad \text{for } b \in \{0,1\},$$

  where $(i^*, i, v) \in [N] \times [0, N] \times \{0,1\}$ is encoded in $\{0,1\}^{\lceil \log_2 N \rceil + \lceil \log_2 (N+1) \rceil + 1}$ in the obvious manner. The algorithm runs

  $$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{HE.Gen}(g_0, g_1, N), \qquad \mathsf{ct} \xleftarrow{\$} \mathsf{HE.Enc}(\mathsf{pk}, (i^*, 0, 0)),$$

  sets $\mathsf{hk} \leftarrow (\mathsf{pk}, \mathsf{ct})$ and $\mathsf{td} \leftarrow \mathsf{sk}$, and outputs $(\mathsf{hk}, \mathsf{td})$.

- $\mathsf{Hash}(\mathsf{hk}, x)$ parses $(\mathsf{pk}, \mathsf{ct})$ from hk. It computes $\mathsf{ct}' \leftarrow \mathsf{HE.Eval}(\mathsf{pk}, x, \mathsf{ct}; 0^\star)$ and outputs ct'. (We fix the randomness of $\mathsf{HE.Eval}$ to the all-zero string. Recall that x specifies the function $g_{x_N} \circ \cdots \circ g_{x_2} \circ g_{x_1}$.)
- $\mathsf{Extract}(\mathsf{td}, h)$ parses sk from td and ct' from h. It runs $(i', i, v) \xleftarrow{\$} \mathsf{HE.Dec}(\mathsf{sk}, \mathsf{ct}')$ and outputs v.

Theorem 11 (¶). *If* HE *is (compact and correct and) selectively secure (Definition 7), then Construction 4 is (correct and) index-hiding (Definition 8).*

Proof (Theorem 11). Write $n = \lceil \log_2 N \rceil + \lceil \log_2 (N+1) \rceil + 1$ for the input length of g_0, g_1, then g_0, g_1, $\overline{T} = N$, and consequently, $\mathsf{param} = (g_0, g_1, \overline{T})$, are of length $\mathrm{poly}(n)$. By the compactness of HE, we have $|\mathsf{ct}'| \le \mathrm{poly}(\lambda, |\mathsf{param}|, n) \le \mathrm{poly}(\lambda, \log N)$, so $|h| < |x|$ once N is beyond some (fixed) polynomial in λ.

By the definitions of g_0, g_1, we have

$$(g_{x_N} \circ \cdots \circ g_{x_2} \circ g_{x_1})(i^*, 0, 0) = (i^*, N, x_{i^*}).$$

By the correctness of HE, the algorithm Extract always outputs the correct value.

The index-hiding property is just the selective security of HE, which says

$$\{(\mathsf{pk}, \mathsf{HE.Enc}(\mathsf{pk}, (i_0^*, 0, 0)))\} \approx \underbrace{\{(\mathsf{pk}, \mathsf{HE.Enc}(\mathsf{pk}, (i_1^*, 0, 0)))\}}_{\mathsf{hk}}$$

for adversarially chosen param, i_0^*, i_1^*. □

7 New Notions for Trapdoor/Pseudorandom Permutations

This section provides more abstract formulations of the properties satisfied by the constructions leading to our schemes. We define a family of cryptographic primitives that exhibit a combination of "white-box" security properties akin to trapdoor permutations, together with "black-box" security properties akin to pseudorandom permutations. We then demonstrate that *i)* these primitives are realized by our constructions and *ii)* they suffice, in and of themselves, for constructing the desired objects — CCA-secure encryption and key encapsulation schemes.

Section 7.1 first defines a cryptographic primitive capturing the security property in Lemma 5 without being tied to a particular instantiation. It then discusses two possible *weakenings* of this primitive, puncturable trapdoor permutations with multiple modes of puncturing, that are still sufficient for (fully adaptive) CCA-secure PKE, plus a third weakening that is provably insufficient for CCA2 security.

Section 7.2 defines a primitive generalizing the security property in Lemma 7 (puncturable trapdoor permutations with cohardcore functions), again without being tied to a particular instantiation.

Section 7.3 abstracts the security property in Lemma 9: puncturable pseudorandom permutations. Here, we also discuss its relation to prior work regarding variants of this notion [BKW17].

7.1 Puncturable TDP with Multiple Modes of Puncturing

We start with an abstraction of Lemma 5, then proceed to the discussion of weakened versions.

Definition 9 (pre-pre-post-puncturable trapdoor permutation). A *112-puncturable TDP*[16] consists of five efficient algorithms.

- $\mathsf{Gen}(1^\lambda, 1^n)$ takes the payload length n as input. It outputs a pair $(\mathsf{pk}, \mathsf{td})$.
- $\mathsf{Eval}(1^\lambda, \mathsf{pk}, x)$ takes as input pk and $x \in \{0,1\}^{n+\lambda}$. It is deterministic and outputs some $y \in \{0,1\}^{n+\lambda}$.
- $\mathsf{Eval}^{-1}(1^\lambda, \mathsf{td}, y)$ takes as input (normal or pre-punctured) td and y. It is supposed to recover x.
- $\mathsf{Puncture}_1(1^\lambda, \mathsf{td}, z)$ takes as input td and $z \in \{0,1\}^\lambda$ (puncturing by randomness) or $z \in \{0,1\}^{n+\lambda}$ (puncturing by image). It outputs a pre-punctured $\mathsf{td}_{1,z}$.
- $\mathsf{Puncture}_2(1^\lambda, \mathsf{td}, y)$ takes as input td and $y \in \{0,1\}^{n+\lambda}$. It outputs a post-punctured $\mathsf{td}_{2,y}$.
- $\mathsf{PayloadEval}^{-1}(1^\lambda, \mathsf{td}, y)$ takes as input a post-punctured td and y. It is supposed to recover the first n bits of x.

The scheme must be correct, i.e., for all $\lambda, n \in \mathbb{N}$, $v, v' \in \{0,1\}^n$, $r, r' \in \{0,1\}^\lambda$, let

$$\begin{aligned}
(\mathsf{pk}, \mathsf{td}) &\xleftarrow{\$} \mathsf{Gen}(1^\lambda, 1^n), & \mathsf{td}_{1,r} &\xleftarrow{\$} \mathsf{Puncture}_1(1^\lambda, \mathsf{td}, r),\\
y &\leftarrow \mathsf{Eval}(1^\lambda, \mathsf{pk}, v\|r), & \mathsf{td}_{1,y} &\xleftarrow{\$} \mathsf{Puncture}_1(1^\lambda, \mathsf{td}, y),\\
y' &\leftarrow \mathsf{Eval}(1^\lambda, \mathsf{pk}, v'\|r'), & \mathsf{td}_{2,y} &\xleftarrow{\$} \mathsf{Puncture}_2(1^\lambda, \mathsf{td}, y),
\end{aligned}$$

then

$$\begin{aligned}
&\Pr\left[\mathsf{Eval}^{-1}(1^\lambda, \mathsf{td}, y) = v\|r\right], && \Pr\left[(r = r') \vee \left(\mathsf{Eval}^{-1}(1^\lambda, \mathsf{td}_{1,r}, y') = v'\|r'\right)\right],\\
& && \Pr\left[(v'\|r' = v\|r) \vee \left(\mathsf{Eval}^{-1}(1^\lambda, \mathsf{td}_{1,y}, y') = v'\|r'\right)\right],\\
& \text{and} && \Pr\left[(v'\|r' = v\|r) \vee \left(\mathsf{PayloadEval}^{-1}(1^\lambda, \mathsf{td}_{2,y}, y') = v'\right)\right]
\end{aligned}$$

are all equal to 1.

The scheme is *112-secure*, if $\mathsf{Exp}^0_{\mathrm{ptdp},112} \approx \mathsf{Exp}^1_{\mathrm{ptdp},112}$, where $\mathsf{Exp}^\beta_{\mathrm{ptdp},112}(1^\lambda, \mathcal{A})$ proceeds as follows (cf. Lemma 5).

- **Setup/Query I.** Launch $\mathcal{A}(1^\lambda)$ and receive 1^n from it. Run/sample

$$\begin{aligned}
(\mathsf{pk}, \mathsf{td}) &\xleftarrow{\$} \mathsf{Gen}(1^\lambda, 1^n), & r &\xleftarrow{\$} \{0,1\}^\lambda, \qquad y_1 \xleftarrow{\$} \{0,1\}^{n+\lambda},\\
\mathsf{td}_{1,0} &\xleftarrow{\$} \mathsf{Puncture}_1(1^\lambda, \mathsf{td}, r), & \mathsf{td}_{1,1} &\xleftarrow{\$} \mathsf{Puncture}_1(1^\lambda, \mathsf{td}, y_1).
\end{aligned}$$

  Send $(\mathsf{pk}, \mathsf{td}_{1,\beta})$ to $\mathcal{A}$.

- **Challenge/Query II.** $\mathcal{A}$ chooses $v^* \in \{0,1\}^n$. Set $y_0 \leftarrow \mathsf{Eval}(1^\lambda, \mathsf{pk}, v^*\|r^*)$ and run $\mathsf{td}_2 \xleftarrow{\$} \mathsf{Puncture}_2(1^\lambda, \mathsf{td}, y_\beta)$. Send (y_β, td_2) to $\mathcal{A}$.
- **Guess.** $\mathcal{A}$ outputs a bit β', which is the output of the experiment.

[16] "112" is short for "pre-pre-post" and refers to Query I, I, and II (pre-/pre-/post-challenge queries), during which punctured trapdoors are given to the adversary.

It is straight-forward to see that *i)* a 112-puncturable TDP gives rise to a CCA-secure PKE, without requiring obfuscation *outside of* the TDP, and *ii)* Lemma 5 can be cast as such a TDP.

Definition 9 is somewhat unnatural — there are three different ways to puncture a trapdoor (by randomness for full preimage; by image for full preimage; by image for payload). It is natural to ask which mode(s) of puncturing can be removed without losing CCA-secure PKE. Observe that puncturing by randomness cannot be removed because the exact punctured point is not determined until Challenge. It turns out that either of the other two can be removed while retaining CCA2 security, but removing both downgrades the straight-forward PKE to CCA1 security.

Definition 10 (pre-post-puncturable trapdoor permutation). The definition of *12-puncturable TDP* is modified from that of 112-puncturable TDP (Definition 9) as follows.

- $\mathsf{Puncture}_1$ only needs to accept $z \in \{0,1\}^\lambda$.
- The correctness requirement on $\mathsf{td}_{1,y}$ is removed.
- In $\mathsf{Exp}^{\beta}_{\mathsf{ptdp},12}$, change the components to

$$\cancel{y_1 \xleftarrow{\$} \{0,1\}^{n+\lambda}} \qquad y_1 \leftarrow \mathsf{Eval}(1^\lambda, \mathsf{pk}, u^* \| r^*) \text{ with } u^* \xleftarrow{\$} \{0,1\}^n,$$
$$\cancel{\mathsf{td}_{1,1} \xleftarrow{\$} \mathsf{Puncture}_1(1^\lambda, \mathsf{td}, y_1)} \qquad \mathsf{td}_{1,1} \leftarrow \mathsf{td}_{1,0}.$$

Definition 10 reduces the modes of pre-challenge puncturing from two to one.

A 12-puncturable TDP still gives rise to a CCA-secure PKE. When designing the hybrids for the PKE version of Theorem 3, in the final hybrid, continue to reject any pre-challenge decryption queries that share encryption randomness with the challenge.

Definition 11 (pre-pre-puncturable trapdoor permutation). The definition of *11-puncturable TDP* is modified from that of 112-puncturable TDP (Definition 9) as follows.

- $\mathsf{Puncture}_2$, $\mathsf{PayloadEval}^{-1}$, and their correctness requirement are removed.
- In $\mathsf{Exp}^{\beta}_{\mathsf{ptdp},11}$, remove td_2.

Definition 11 does away with post-challenge puncturing. This is arguably more natural as td_2 (unnaturally) only allows recovery of payload, not the randomness part of the preimage. Note that the two modes of pre-challenge puncturing are required.

A 11-puncturable TDP still gives rise to a CCA-secure PKE. This is explained in Remark 2. Basically, the fact that $\mathsf{td}_{1,0}$ can be switched to $\mathsf{td}_{1,1}$ implies that except with negligible probability, the adversary does not make any query with colliding randomness in the post-challenge phase.

Definition 12 (pre-puncturable trapdoor permutation; does not imply CCA-secure PKE). The definition of *1-puncturable TDP* is modified from that of 112-puncturable TDP (Definition 9) as follows.

- $\mathsf{Puncture}_1$ only needs to accept $z \in \{0,1\}^\lambda$.
- $\mathsf{Puncture}_2$ and $\mathsf{PayloadEval}^{-1}$ are removed.

- The correctness requirements on $\mathsf{td}_{1,y}$, td_2 are removed.
- In $\mathsf{Exp}^{\beta}_{\mathsf{ptdp},1}$, change the components to

$$\begin{aligned} &\cancel{y_1 \xleftarrow{\$} \{0,1\}^{n+\lambda}} && y_1 \leftarrow \mathsf{Eval}(1^\lambda, \mathsf{pk}, u^* \| r^*) \text{ with } u^* \xleftarrow{\$} \{0,1\}^n, \\ &\cancel{\mathsf{td}_{1,1} \xleftarrow{\$} \mathsf{Puncture}_1(1^\lambda, \mathsf{td}, y_1)} && \mathsf{td}_{1,1} \leftarrow \mathsf{td}_{1,0}, \\ &\cancel{\mathsf{td}_2 \xleftarrow{\$} \mathsf{Puncture}_2(1^\lambda, \mathsf{td}, y_\beta)} && \mathsf{td}_2 \leftarrow \bot. \end{aligned}$$

Definition 12 is a weakened version combining Definitions 10 and 11. The straight-forward PKE is CCA1-secure but it can be CCA2-*insecure*, such as in the following example. Effectively, the counterexample is hybrid encryption using CCA-secure KEM and (CCA-insecure) OTP.

Ingredients of Construction 5. Let

- $\mathsf{PPRP} = (\mathsf{PPRP.Gen}, \mathsf{PPRP.Permute}, \mathsf{PPRP.Eval}, \mathsf{PPRP.Eval}^{-1})$ be an IP-PRP,
- $F : \{0,1\}^\lambda \times \mathbb{N} \to \{0,1\}$ a PRF, and
- $i\mathcal{O}$ a circuit obfuscator.

Construction 5 (pre-puncturable TDP failing CCA2 security). It works as follows.

- $\mathsf{Gen}(1^n)$ runs

  $$\pi \xleftarrow{\$} \mathsf{PPRP.Gen}(1^\lambda), \quad \widetilde{\pi} \xleftarrow{\$} i\mathcal{O}(\pi), \quad \sigma \xleftarrow{\$} \mathsf{PPRP.Gen}(1^\lambda), \quad \widetilde{\sigma} \xleftarrow{\$} i\mathcal{O}(\sigma).$$

  It sets $\mathsf{pk} \leftarrow (\widetilde{\pi}, \widetilde{\sigma})$, $\mathsf{td} \leftarrow (\pi^{-1}, \widetilde{\sigma})$, and outputs $(\mathsf{pk}, \mathsf{td})$.
- $\mathsf{Eval}(\mathsf{pk}, x)$ parses pk into $(\widetilde{\pi}, \widetilde{\sigma})$ and x into $v \| r$ with $r \in \{0,1\}^\lambda$. The algorithm runs $u_i \leftarrow F(\widetilde{\sigma}(r), i)$ for $i \in [n]$ and outputs $y \leftarrow ((u_1 \| \cdots \| u_n) \oplus v) \| \widetilde{\pi}(r)$.
- $\mathsf{Eval}^{-1}(\mathsf{td}, y)$ parses y into $w \| s$ with $s \in \{0,1\}^\lambda$. If $\mathsf{td} = (\pi^{-1}, \widetilde{\sigma})$ is not punctured, it computes $r \leftarrow \pi^{-1}(s)$; otherwise, $\mathsf{td} = (\widetilde{Q}, \widetilde{\sigma})$ is punctured, and the algorithm runs $r \leftarrow \widetilde{Q}(s)$. In either case, it proceeds with $u_i \leftarrow F(\widetilde{\sigma}(r), i)$ for $i \in [n]$ and outputs $x \leftarrow ((u_1 \| \cdots \| u_n) \oplus w) \| r$.
- $\mathsf{Puncture}_1(\mathsf{td}, z)$ parses td into $(\pi^{-1}, \widetilde{\sigma})$. The algorithm runs $\widetilde{Q} \xleftarrow{\$} i\mathcal{O}(\pi^{-1} \setminus \{\pi(z)\})$ and outputs $\mathsf{td}_{1,z} \leftarrow (\widetilde{Q}, \widetilde{\sigma})$.

Corollary 12 (of Theorem 6). *Suppose $i\mathcal{O}$, PPRP are secure (Definitions 1 and 2), PPRP supports all transpositions (see Lemma 1), and F is a secure PRF (the standard definition), then Construction 5 is 1-secure (Definition 12).*

When Construction 5 is used as a PKE in the straight-forward manner, it is not CCA2-secure, as the payload part of a ciphertext is easily malleable. The reduction (from breaking pre-puncturable TDP to breaking CCA2 security) fails because a successful adversary against CCA2 security will query decryption on a ciphertext that *always* shares encryption randomness with the challenge one, which cannot be answered by the reduction.

Remark 4 (relation with adaptive TDF [KMO10]). Adaptive TDF are trapdoor functions that remain one-way (for random images) even when the adversary is given adaptive access to an inversion oracle punctured at the challenge image. They are introduced to capture needed properties for obtaining CCA-secure PKE from TDF. The direct construction of PKE from adaptive TDF is bit encryption, which must then be converted for multi-bit messages via [Ms09].

Our notion strengthens several aspects of adaptive TDF: security is pseudorandomness instead of just one-wayness; the preimage is partially adversarially controlled; the punctured inversion oracle is replaced by a punctured trapdoor (code access). Indeed, trapdoor permutations that satisfy our notion can be used as PKE in a more straight-forward way. In particular, the scheme supports multi-bit messages with a single application of of the TDP.

7.2 Puncturable TDP with Cohardcore Functions for Distorted Input

We define a notion of puncturable trapdoor permutations (only a single mode of puncturing here) and a general, strong security property, and then prove it for the trapdoor permutation of [SZ25] — it is secure for many *distorted input* distributions, has a broad class of *cohardcore functions*, and satisfies *white-box CCA* security. Let us elaborate these points.

Distorted Input. One-wayness is usually defined for uniformly random input. Existing studies of security against distorted input without additional randomness focus on distributions with high min-entropy [BBO07,PW08,BFO08,BFOR08] and rely on either random oracles or lossy injective functions. Our result holds when the input has sufficient extracted false entropy, subject to some other technical requirements.[17]

Cohardcore Functions. Recall that h is hardcore for f if $h(x)$ is pseudorandom given $f(x)$, with the interesting case being when $f(x)$ determines $h(x)$. In contrast, h is *cohardcore* for f if $f(x)$ is pseudorandom given $h(x)$, with the interesting case *still* being when $f(x)$ determines $h(x)$. If we only consider pseudorandomness, h being cohardcore for f is the same as f being hardcore for h. The choice of terminology emphasizes *which function determines which*.

As an example, if a PKE scheme has pseudorandom ciphertexts, the plaintext is cohardcore for the ciphertext. Of course, if h has marginally pseudorandom images and is cohardcore for a permutation f, then it is hardcore for f.

Our result proves the cohardcoreness of a broad class of functions.

White-Box CCA (Puncturability). Adaptive trapdoor functions [KMO10] are those that remain one-way even in the presence of an inversion oracle punctured at the target image. They naturally give rise to CCA-secure PKE/KEM. In our result, the adversary is given a punctured inverse (obfuscated program), which is a stronger than oracle access.

Putting Things Together. The aforementioned properties hold jointly: a broad class of functions h are cohardcore for the trapdoor permutation of [SZ25] (i.e., $i\mathcal{O}(\pi)$), even

[17] True entropy can be extracted from high-min-entropy distributions by universal hashing, so our condition appears weaker. Alas, additional technical requirements make the two incomparable.

in the presence of a punctured inverse $i\mathcal{O}(\pi^{-1} \setminus \{\pi(x^*)\})$, as long as the preimage (x^*) distribution has sufficient false entropy explicitly extracted by some h' and x^* is uniquely determined by (h, h').

Definition 13 (puncturable trapdoor permutation). A *puncturable TDP* consists of four efficient algorithms.

- $\mathsf{Gen}(1^\lambda, 1^n)$ takes the domain length n as input. It outputs a pair $(\mathsf{pk}, \mathsf{td})$.
- $\mathsf{Eval}(1^\lambda, \mathsf{pk}, x)$ takes as input pk and $x \in \{0,1\}^n$. It is deterministic and outputs some $y \in \{0,1\}^n$.
- $\mathsf{Eval}^{-1}(1^\lambda, \mathsf{td}, y)$ takes as input (punctured or not) td and $y \in \{0,1\}^n$. It is supposed to recover x.
- $\mathsf{Puncture}(1^\lambda, \mathsf{td}, y')$ takes as input (non-punctured) td and $y' \in \{0,1\}^n$. It outputs a punctured trapdoor td'.

The scheme must be *correct*, i.e., for all $\lambda, n \in \mathbb{N}$, $x, y', y'' \in \{0,1\}^n$, let

$$(\mathsf{pk}, \mathsf{td}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda, 1^n), \qquad y \leftarrow \mathsf{Eval}(1^\lambda, \mathsf{pk}, x), \qquad \mathsf{td}' \xleftarrow{\$} \mathsf{Puncture}(1^\lambda, \mathsf{td}, y'),$$

then

$$\Pr\left[\mathsf{Eval}^{-1}(1^\lambda, \mathsf{td}, y) = x\right] \quad \text{and} \quad \Pr\left[(y' = y'') \vee \left(\mathsf{Eval}^{-1}(1^\lambda, \mathsf{td}', y'') = \mathsf{Eval}^{-1}(1^\lambda, \mathsf{td}, y'')\right)\right]$$

are equal to 1.

Definition 14 (pseudorandomness, cohardcore functions, distorted input). Let $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Eval}^{-1}, \mathsf{Puncture})$ be a puncturable TDP scheme (Definition 13). Let $\mathcal{D}$ be a *sampler* of syntax $(h, x^*, \mathsf{aux}') \xleftarrow{\$} \mathcal{D}(1^\lambda)$, where $h : \{0,1\}^{|x^*|} \to \{0,1\}^m$ is a circuit and x^*, aux' are strings. The scheme is *$\mathcal{D}$-pseudorandom* if

$$\left\{\left(h,\ \mathsf{aux}',\ h(x^*),\ \boxed{y_0}\ ,\mathsf{pk},\ \boxed{\mathsf{td}'_0}\right)\right\} \approx \left\{\left(h,\ \mathsf{aux}',\ h(x^*),\ \boxed{y_1}\ ,\mathsf{pk},\ \boxed{\mathsf{td}'_1}\right)\right\},$$

where

$$(\mathsf{pk}, \mathsf{td}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda, 1^{|x^*|}), \qquad y_0 \leftarrow \mathsf{Eval}(1^\lambda, \mathsf{pk}, x^*), \qquad y_1 \xleftarrow{\$} \{0,1\}^n,$$
$$\mathsf{td}'_\beta \xleftarrow{\$} \mathsf{Puncture}(1^\lambda, \mathsf{td}, y_\beta) \quad \text{for } \beta \in \{0,1\}.$$

In this case, h is said to be its *cohardcore* function against the distribution of x^* with auxiliary information aux' specified by $\mathcal{D}$.

Many Samplers, But Not All. A puncturable TDP is never $\mathcal{D}$-pseudorandom if x^* can be efficiently guessed from h, aux, $h(x^*)$. We will show that the trapdoor permutation of [SZ25] has cohardcore function h against for arbitrary x^* and auxiliary information, subject to some technical condition about h and false entropy of x^*.

Ingredients of Construction 6. Let

- $\mathsf{PPRP} = (\mathsf{PPRP.Gen}, \mathsf{PPRP.Permute}, \mathsf{PPRP.Eval}, \mathsf{PPRP.Eval}^{-1})$ be an IP-PRP, and
- $i\mathcal{O}$ a circuit obfuscator.

Construction 6 (puncturable TDP with cohardcore functions for distorted input). It works as follows.

- $\mathsf{Gen}(1^n)$ samples a PPRP instance $\pi \xleftarrow{\$} \mathsf{PPRP.Gen}(1^n)$, obfuscates its forward evaluation into $\mathsf{pk} \xleftarrow{\$} i\mathcal{O}(\pi)$, and sets $\mathsf{td} \leftarrow \pi^{-1}$. The algorithm outputs $(\mathsf{pk}, \mathsf{td})$.
- $\mathsf{Eval}(\mathsf{pk}, x)$ runs and outputs $\mathsf{pk}(x)$.
- $\mathsf{Eval}^{-1}(\mathsf{td}, y)$ checks the format of td. If $\mathsf{td} = \pi^{-1}$, it evaluates and outputs $\pi^{-1}(y)$; otherwise, td is a circuit, and the algorithm runs and outputs $\mathsf{td}(y)$.
- $\mathsf{Puncture}(\mathsf{td}, y)$ runs and outputs $\mathsf{td}_y \xleftarrow{\$} i\mathcal{O}(\pi^{-1} \setminus \{y\})$.

The paddings in the two calls to $i\mathcal{O}$ grow with certain parameters of $\mathcal{D}$.

Theorem 13 (¶). *Suppose $i\mathcal{O}$, PPRP are secure (Definitions 1 and 2) and PPRP supports all transpositions (see Lemma 1). Suppose $\mathcal{D}$ is a sampler (Definition 14). If it satisfies the following conditions (for $(h, x^*, \mathsf{aux}') \xleftarrow{\$} \mathcal{D}()$)...*

- *The auxiliary information aux' is in the format of (h', aux), where aux is a string and $h' : \{0,1\}^{|x^*|} \to \{0,1\}^{m'}$ is a circuit.*
- *The preimage x^* has no sibling under (h, h'), i.e.,*

$$\Pr\left[(h(x), h'(x)) \neq (h(x^*), h'(x^*)) \text{ for all } x \in \{0,1\}^{|x^*|} \setminus \{x^*\}\right] = 1.$$

- *The codomain of h' is large, i.e., $\Pr[m' \geq \lambda] = 1$.*
- *The value $h'(x^*)$ is pseudorandom given $h(x^*)$ and other data, i.e., for $z' \xleftarrow{\$} \{0,1\}^{m'}$,*

$$\left\{(h, h', \mathsf{aux}, h(x^*), \boxed{h'(x^*)})\right\} \approx \left\{(h, h', \mathsf{aux}, h(x^*), \boxed{z'})\right\}.$$

- *The circuits being obfuscated (in the scheme) are padded to at least some particular $\Theta(|h| + |h'|)$ size.*

... then Construction 6 is $\mathcal{D}$-pseudorandom (Definition 14).

The output length of $\mathcal{D}$ should be polynomial.[18] *The proof uses slightly different reductions depending on its desired (non-)uniformity. Uniform ones run in roughly the time of $\mathcal{D}$ plus the distinguisher, so the time of $\mathcal{D}$ must be bearable for $i\mathcal{O}$, PPRP. Non-uniform ones run in roughly the time of the distinguisher, when given a sample from $\mathcal{D}()$ as advice, so $\mathcal{D}$ need not be efficient. The advantage is $\varepsilon_{\text{chcf}} = 7\varepsilon_{i\mathcal{O}} + 2\varepsilon_{\text{trigger}} + \varepsilon_{\text{PPRP}} + \varepsilon_{\mathcal{D}}$, where $\varepsilon_{\mathcal{D}}$ relates to the premise of pseudorandomness.*

Technically, since (h, h', aux) could be arbitrarily correlated with x^*, the cohardcore function value should be thought as $(h, h', \mathsf{aux}, h(x^*))$ instead of just $h(x^*)$.

[18] For subexponential security, this is still required, as otherwise the distinguisher would be allowed superexponential time.

Applications. Before proving Theorem 13, we first present two applications.

Hardcore Bits for the Trapdoor Permutation of [SZ25]. Let $\mathcal{D}()$ choose $n \le \mathrm{poly}(\lambda)$ and $S \subseteq [n]$ with $|S| \le n - \lambda$, define $h(x) := x|_S$, $h'(x) := x|_{[n]\setminus S}$, sample $x^* \xleftarrow{\$} \{0,1\}^n$, and set $\mathsf{aux} \leftarrow (1^n, S)$. It satisfies the premise of Theorem 13 (the tuple map (h, h') is injective, $m' \ge \lambda$, and $h'(x^*)$ is random), so (after rearranging/discarding components)

$$\begin{aligned}\left\{\left(1^n,\, S,\, x^*|_S\,,\, \boxed{\pi(x^*)}\,,\, i\mathcal{O}(\pi)\right)\right\} &\approx \left\{\left(1^n,\, S,\, x^*|_S\,,\, \boxed{\quad y_1 \quad}\,,\, i\mathcal{O}(\pi)\right)\right\} \\ &\equiv \left\{\left(1^n,\, S,\, \boxed{x'}\,,\, \boxed{\pi(x^*)}\,,\, i\mathcal{O}(\pi)\right)\right\},\end{aligned}$$

where $y_1 \xleftarrow{\$} \{0,1\}^n$ and $x' \xleftarrow{\$} \{0,1\}^{|S|}$. In other words, any subset of preimage bits are hardcore as long as sufficiently many preimage bits are dropped.

Recovering Lemma 7 (plaintext-selective CCA with rerandomization by PPRP). This example is interesting in two aspects: *i)* it is obtained by applying Theorem 13 twice, the second time on top of the first; *ii)* $h'(x^*)$ is only pseudorandom and not random at all.

Let $\mathcal{D}()$ sample aux to be the randomness for $\mathcal{A}$ and compute v^* as $\mathcal{A}$ would. It runs and samples

$$\sigma \xleftarrow{\$} \mathsf{Gen}(1^{2+\lambda}), \qquad \widetilde{\sigma} \xleftarrow{\$} i\mathcal{O}(\sigma), \qquad r_1^* \xleftarrow{\$} \{0,1\}^\lambda, \qquad r_2^* \xleftarrow{\$} \{0,1\}^\lambda,$$

and defines and sets

$$h(v\|r_1\|r_2) := v\|\widetilde{\sigma}(r_1\|r_2), \qquad h'(v\|r_1\|r_2) := r_1, \qquad x^* \leftarrow v^*\|r_1^*\|r_2^*.$$

Now that h is already injective by itself, h' has output length λ, and by the hardcore bits result, $h'(x^*) = r_1^*$ is pseudorandom given $h(x^*)$ and other data, we derive (after discarding some components)

$$\begin{aligned}&\left\{\left(\mathsf{aux},\, \overbrace{i\mathcal{O}(\sigma)}^{h},\, \overbrace{\sigma(r_1^*\|r_2^*)}^{h(x^*)},\, \boxed{\pi(v^*\|r_1^*\|r_2^*)}\,,\, i\mathcal{O}(\pi),\, i\mathcal{O}(\pi^{-1}\setminus\{\pi(v^*\|r_1^*\|r_2^*)\})\right)\right\} \\ \approx &\left\{\left(\mathsf{aux},\, i\mathcal{O}(\sigma),\, \underbrace{\sigma(r_1^*\|r_2^*)}_{s_1},\, \boxed{\quad y_1 \quad}\,,\, i\mathcal{O}(\pi),\, i\mathcal{O}(\pi^{-1}\setminus\{\quad y_1 \quad\})\right)\right\}.\end{aligned}$$

The reason we use 2λ is to adhere to the premise of Theorem 13. It can be tuned down to λ without affecting asymptotic security.

Proof. We now prove Theorem 13.

Proof (Theorem 13). The proof follows the usual trigger–permute–untrigger strategy. Let $(\not{\Delta}, \Delta)$ be a trigger scheme (potentially subexponentially secure) promised by Lemma 2 and $\mathcal{T}_1$, $\mathcal{T}_2$ triggers potentially activated at $x' \xleftarrow{\$} \{0,1\}^{|x^*|}$, $z' \xleftarrow{\$} \{0,1\}^{m'}$ (or $h'(x^*)$). In each hybrid, the distinguisher is given (unchanging) h, h', aux, $h(x^*)$, and (changing) y^*, $\mathsf{pk} \xleftarrow{\$} i\mathcal{O}(P)$, $\mathsf{td}'_* \xleftarrow{\$} i\mathcal{O}(Q)$. Consider the following hybrids, specified by y^*, P, Q.

- H_0 is the left distribution, where

$$\begin{aligned} y^* &= \pi(x^*), \\ P(x) &= \pi(x), \end{aligned} \qquad Q(y) = \begin{cases} \bot, & \text{if } y = y^*; \\ \pi^{-1}(y), & \text{otherwise.} \end{cases}$$

- In H_1, we sample $\mathcal{T}_1 \xleftarrow{\$} \not\Delta$ and update the programs:

$$y^* = \pi(x^*),$$
$$P(x) = \begin{cases} \boxed{\pi(x^*),} & \boxed{\text{if } \mathcal{T}_1(x) = 1;} \\ \pi(x), & \text{otherwise;} \end{cases} \qquad Q(y) = \begin{cases} \bot, & \text{if } y = y^*; \\ \boxed{x^*,} & \boxed{\text{else if } \mathcal{T}_1(\pi^{-1}(y)) = 1;} \\ \pi^{-1}(y), & \text{otherwise.} \end{cases}$$

  Since the newly added branches are never entered, $\mathsf{H}_0 \approx \mathsf{H}_1$ by $i\mathcal{O}$ security.

- In H_2, we switch $\mathcal{T}_1$ to $\mathcal{T}_1 \xleftarrow{\$} \boxed{\Delta(x')}$.

  By trigger scheme security, $\mathsf{H}_1 \approx \mathsf{H}_2$.

- In H_3, we run $\pi_* \xleftarrow{\$} \mathsf{Permute}(\pi, (x^* \;\; x'), 0)$ and rewrite everything using π_*, x':

$$y^* = \boxed{\pi_*}(x^*),$$
$$P(x) = \begin{cases} \boxed{\pi_*}(x^*), & \text{if } \boxed{x = x'}; \\ \boxed{\pi_*}(x), & \text{otherwise;} \end{cases} \qquad Q(y) = \begin{cases} \bot, & \text{if } \boxed{\pi_*^{-1}(y) = x^*}; \\ x^*, & \text{else if } \boxed{\pi_*^{-1}(y) = x'}; \\ \boxed{\pi_*^{-1}}(y), & \text{otherwise.} \end{cases}$$

  The value of y^* and the functionalities of P, Q remain unchanged, so $\mathsf{H}_2 \approx \mathsf{H}_3$ by $i\mathcal{O}$ security.

- In H_4, we let $\pi_* \xleftarrow{\$} \mathsf{Permute}(\pi, (x^* \;\; x'), \boxed{1})$.

  By PPRP security, $\mathsf{H}_3 \approx \mathsf{H}_4$.

- In H_5, we let $z \leftarrow h(x^*)$, sample $\mathcal{T}_2 \xleftarrow{\$} \Delta(h'(x^*))$, and rewrite the components to use π, z, $\mathcal{T}_2$ instead of π_*, x^*:

$$y^* = \boxed{\pi(x')},$$
$$\boxed{P}(x) = \begin{cases} \pi(x'), & \text{if } \boxed{h(x) \| \mathcal{T}_2(h'(x)) = z \| 1}; \\ \pi(x), & \text{otherwise;} \end{cases} \qquad \boxed{Q}(y) = \begin{cases} \bot, & \text{if } y = y^*; \\ \pi^{-1}(y), & \text{otherwise.} \end{cases}$$

  By the no-sibling condition, $h(x) \| \mathcal{T}_2(h'(x)) = z \| 1$ if and only if $x = x^*$. It is readily verified that y^* and the functionalities of P, Q are unchanged. Therefore, $\mathsf{H}_4 \approx \mathsf{H}_5$ by $i\mathcal{O}$ security.

- In H_6, we switch $\mathcal{T}_2$ to $\mathcal{T}_2 \xleftarrow{\$} \Delta(\boxed{z'})$.

  $\mathsf{H}_5 \approx \mathsf{H}_6$ by the pseudorandomness of $h'(x^*)$.

- In H_7, we switch $\mathcal{T}_2$ again, to $\mathcal{T}_2 \xleftarrow{\$} \boxed{\not\Delta}$.

  Since $m' \geq \lambda$, by trigger scheme security, $\mathsf{H}_6 \approx \mathsf{H}_7$.

- In H_8, we remove the inactive branch of P:

$$y^* = \pi(x'),$$
$$P(x) = \begin{cases} \sout{\pi(x'),} & \sout{\text{if } h(x) \| \mathcal{T}_2(h'(x)) = z \| 1;} \\ \pi(x), & \sout{\text{otherwise;}} \end{cases} \qquad Q(y) = \begin{cases} \bot, & \text{if } y = y^*; \\ \pi^{-1}(y), & \text{otherwise.} \end{cases}$$

  By $i\mathcal{O}$ security, $\mathsf{H}_7 \approx \mathsf{H}_8$.

Note that H_8 is identical to the right distribution by renaming $\pi(x')$ to y_1. The theorem follows by hybrid argument. □

7.3 Puncturable PRP

We abstract Lemma 9 then revisit and reconciliate with [BKW17].

Definition 15 (puncturable pseudorandom permutation). A *puncturable PRP* consists of four efficient algorithms.

- $\mathsf{Gen}(1^\lambda, 1^n)$ takes the domain length n as input. It outputs a (non-punctured) key k.
- $\mathsf{Puncture}(1^\lambda, k, x, y)$ takes as input a non-punctured key k, a PRP input $x \in \{0,1\}^n$, and a PRP output $y \in \{0,1\}^n$. It outputs a key k° punctured at (x, y).
- $\mathsf{Eval}(1^\lambda, k, x)$ takes as input (punctured or not) k and $x \in \{0,1\}^n$. It is deterministic and outputs some y.
- $\mathsf{Eval}^{-1}(1^\lambda, k, y)$ takes as input (punctured or not) k and y. It outputs some x.

The scheme must be *correct*, i.e., for all $\lambda, n \in \mathbb{N}$, $x, x', y' \in \{0,1\}^n$, let

$$k \xleftarrow{\$} \mathsf{Gen}(1^\lambda, 1^n), \qquad y \leftarrow \mathsf{Eval}(1^\lambda, k, x), \qquad k^\circ \xleftarrow{\$} \mathsf{Puncture}(1^\lambda, k, x, y),$$

then

$$\begin{aligned} & \Pr\left[(y \in \{0,1\}^n) \wedge \left(\mathsf{Eval}^{-1}(1^\lambda, k, y) = x\right)\right], \\ & \Pr\left[(x = x') \vee \left(\mathsf{Eval}(1^\lambda, k^\circ, x') = \mathsf{Eval}(1^\lambda, k, x')\right)\right], \\ \text{and} \quad & \Pr\left[(y = y') \vee \left(\mathsf{Eval}^{-1}(1^\lambda, k^\circ, y') = \mathsf{Eval}^{-1}(1^\lambda, k, y')\right)\right], \end{aligned}$$

are all equal to 1.

The scheme is *secure* if $\mathsf{Exp}^0_{\mathrm{pnctr}} \approx \mathsf{Exp}^1_{\mathrm{pnctr}}$, where $\mathsf{Exp}^\beta_{\mathrm{pnctr}}(1^\lambda, \mathcal{A})$ proceeds as follows (cf. Lemma 9).

- **Challenge.** Launch $\mathcal{A}(1^\lambda)$ and receive $x^* \in \{0,1\}^\star$ from it. Run

$$k \xleftarrow{\$} \mathsf{Gen}(1^\lambda, 1^{|x^*|}), \quad y_0 \leftarrow \mathsf{Eval}(1^\lambda, k, x^*), \qquad y_1 \xleftarrow{\$} \{0,1\}^{|x^*|},$$
$$k_0^\circ \xleftarrow{\$} \mathsf{Puncture}(1^\lambda, k, x^*, y_0), \quad k_1^\circ \xleftarrow{\$} \mathsf{Puncture}(1^\lambda, k, x^*, y_1).$$

  Send (y_β, k_β°) to $\mathcal{A}$.

- **Guess.** $\mathcal{A}$ outputs a bit β'. The output of the experiment is β' if $|x^*| \geq \lambda$; otherwise, the output is set to 0.

Requiring large domain for security is justified in Footnote 14. We note that when k is punctured at (x, y) for $y \neq \mathsf{Eval}(k, x)$, its functionality is computationally retained due to security, but this may create issues when used in conjunction with $i\mathcal{O}$. In particular, the proof of Claim 10 (hence that of Theorem 8) relies on perfect correctness for the case of $y \neq \mathsf{Eval}(k, x)$. The definition can be strengthened in that aspect.

Clearly, Lemma 9 gives a construction for Definition 15 (with strong correctness).

Remark 5 (reconciliation with [BKW17]). Lemma 9 "punctures" a PRP per Definition 15, but the work of [BKW17] shows that a PRP cannot be securely "punctured". Upon closer examination, the definition of secure puncturing in [BKW17] is not amenable to permutations. There, a key punctured at x^* supports bidirectional evaluation, and in particular, inversion is allowed on any point except $\mathsf{Eval}(k, x^*)$. Such a key enables easy distinguishing between the function value $\mathsf{Eval}(k, x^*)$ and a random value, trivially by whether inversion succeeds at this value. We consider this a deficiency of definition/modeling. Instead, the right notion should ensure that the point with forbidden inversion is always the challenge value.

8 Instantiations with Reversible Circuits

In this section, we present simple constructions of PPRP, CCA-secure KEM, and (plaintext-selective) CCA-secure PKE using reversible circuits. They are proven secure assuming $i\mathcal{O}$ for reversible circuits and the split-circuit pseudorandomness (SCP) assumption of [CCMR24]. The KEM/PKE scheme might be practical when instantiated with the local-mixing obfuscator of [CCMR24].

8.1 Section Preliminaries

For each $n \in \mathbb{N}_{\geq 3}$, a *(base) gate* (on n wires) is a Toffoli gate ξ specified by (f, w_1, w_2, w_3), where $f : \{0,1\}^2 \to \{0,1\}$ is the control function and $w_1, w_2, w_3 \in [n]$ are distinct wire indices. We conflate the description of a gate with its action, which is

$$\xi(x_1 \| \cdots \| x_n) = y_1 \| \cdots \| y_n, \qquad y_i = \begin{cases} x_i, & \text{if } i \neq w_3; \\ x_{w_3} \oplus f(x_{w_1}, x_{w_2}), & \text{if } i = w_3; \end{cases}$$

for $x_1, \ldots, x_n \in \{0,1\}$. Denote by $\mathbb{B}_n$ the set of all (base) gates on n wires.

Define $\mathcal{C}_{n,m} := \mathbb{B}_n^m$ to be the set of reversible circuits with n wires and m gates and $\mathcal{C}_{n,\star} := \mathbb{B}_n^\star = \bigcup_{m \in \mathbb{N}} \mathcal{C}_{n,m}$ without gate number constraint. Each $C \in \mathcal{C}_{n,m}$ is described by $\xi_1 \| \cdots \| \xi_m$ with $\xi_i \in \mathbb{B}_n$ being the ith gate of C, and it computes the permutation $\xi_m \circ \cdots \circ \xi_1$. It can be depicted as in Figure 1. Write $|C| = m$ for the number of gates in C so that $C \in \mathcal{C}_{n,|C|}$. Let C^{-1} be the literal inverse of C, i.e., $\xi_m \| \cdots \| \xi_1$. Since each gate computes its own inverse, C^{-1} computes the inverse of the permutation computed by C. We abuse the notation so that C is both its description and the permutation computed by it. For $C_1, C_2 \in \mathcal{C}_{n,\star}$, denote by $C_1 \| C_2$ their concatenation, which is a

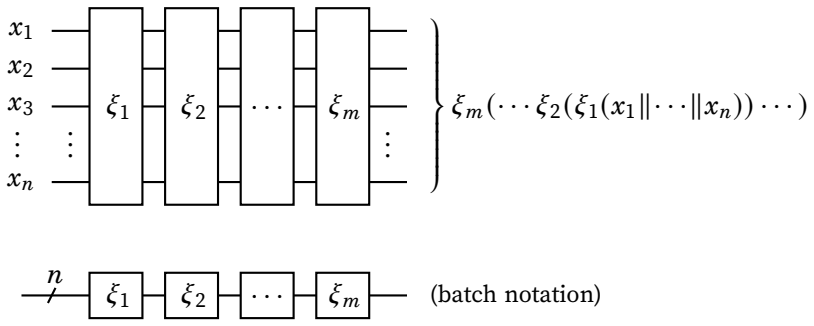


Figure 1. A reversible circuit $C = \xi_1 \| \cdots \| \xi_m$. Gates are applied from left to right.

$(|C_1| + |C_2|)$-gate circuit and computes $C_2 \circ C_1$. For a constant reminder that "$\|$" and "$\circ$" work backwards to each other, we write

$$C_2 \circ C_1 =: C, \qquad (\leftarrow \text{read this way})$$
$$(\text{read this way} \rightarrow) \qquad C := C_1 \| C_2,$$

when defining a new circuit (and = when describing one).

Reversible Circuits for Even Permutations of Short Cycle Decomposition. For $n \geq 4$, any permutation computed by a reversible circuit must be *even*, since each gate in $\mathbb{B}_n$ computes an even permutation. Conversely, every even permutation can be computed by a reversible circuit. The following provides an upper bound on its complexity by its cycle decomposition.

Lemma 14 ([Bro04; Theorem 1]). *There is an efficient deterministic algorithm that takes as input 1^n (with $n \geq 3$) and the cycle representation σ of an even permutation over $\{0,1\}^n$, outputs a reversible circuit C with n wires and $O(|\sigma| \cdot n)$ gates that computes σ. (Trivial cycles do not contribute to $|\sigma|$.)*

Obfuscating Reversible Circuits. One can consider a version of $i\mathcal{O}$ that preserves the format of reversible circuits.

Definition 16 (reversible-circuit $i\mathcal{O}$). A *reversible-circuit obfuscator* is an efficient algorithm $\mathrm{rc}i\mathcal{O}(1^\lambda, C)$ transforming a *reversible* circuit $C \in \mathcal{C}_{n,\star}$ into *another reversible* circuit $\widetilde{C} \in \mathcal{C}_{n,\star}$ computing the same permutation whenever $n \geq \max\{4, \lambda\}$, i.e., for all such C and $x \in \{0,1\}^n$, it holds that $\Pr[\mathrm{rc}i\mathcal{O}(1^\lambda, C)(x) = C(x)] = 1$. The *indistinguishability-based security* is $\mathsf{Exp}^0_{\mathrm{io}} \approx \mathsf{Exp}^1_{\mathrm{io}}$ (Definition 1) except that C_0, C_1 must be both from $\mathcal{C}_{n,m}$ for some $n \geq \max\{4, \lambda\}$ and m.

Remark 6 (format-preserving nature of $\mathrm{rc}i\mathcal{O}$*).* Let C be a reversible circuit. Importantly, $\mathrm{rc}i\mathcal{O}(C)$ allows efficient evaluation of C^{-1}, but $i\mathcal{O}(C)$ may not. As we shall see later, assuming $i\mathcal{O}$ (or $\mathrm{rc}i\mathcal{O}$) and under Assumption 1 (SCP), $i\mathcal{O}(C)$ for a random C (with enough wires, gates, and padding) *prevents* efficient inversion of C, because $\mathrm{rc}i\mathcal{O}(C)$ (or $(i\mathcal{O}(C), i\mathcal{O}(C^{-1}))$) is a PPRP (Theorem 16).

The SCP Assumption. Given $n, m, m' \in \mathbb{N}$ (with $n \geq 3$ and $m' \geq m$) and $C \in \mathcal{C}_{n,m}$, let $\mathcal{E}_{C,m'}$ be the uniform distribution over all n-wire m'-gate reversible circuits that compute the same permutation as C, i.e., $\mathcal{E}_{C,m'}$ picks a uniformly random element of the set

$$\left\{ C' \in \mathcal{C}_{n,m'} \,\middle|\, C' \text{ computes the same permutation as } C \right\}.$$

The distribution $\mathcal{E}_{C,m'}$ can be regarded as an (inefficient) perfect $\mathrm{rc}i\mathcal{O}$ of C with the output circuit size (number of gates) fixed to m'. We recall the following assumption from [CCMR24] regarding the nature of reversible circuits, which they use to construct $\mathrm{rc}i\mathcal{O}$ for all circuits from $\mathrm{rc}i\mathcal{O}$ for short random circuits.

Assumption 1 (split-circuit pseudorandomness [CCMR24]). For some polynomials $m(\lambda, n)$, $m^\#(\lambda, n)$ with integer coefficients such that $m(\lambda, n), m^\#(\lambda, n) > 0$ as long as $n \geq \lambda \geq 0$, it holds that $\mathsf{Exp}^0_{\mathrm{scp}} \approx \mathsf{Exp}^1_{\mathrm{scp}}$, where $\mathsf{Exp}^\beta_{\mathrm{scp}}(1^\lambda, \mathcal{A})$ proceeds as follows.

- **Challenge.** Launch $\mathcal{A}(1^\lambda)$ and receive from it 1^n, $1^{m'_1}$, $1^{m'_2}$, $C_1, C_2 \in \mathcal{C}_{n,\star}$. If

$$n < \max\{4, \lambda\} \quad \text{or} \quad m'_1 < (|C_1| + m(\lambda, n)) \cdot m^{\#}(\lambda, n) \quad \text{or} \quad m'_2 < (m(\lambda, n) + |C_2|) \cdot m^{\#}(\lambda, n),$$

  then output 0 and terminate. Otherwise, sample and set

$$R \xleftarrow{\$} \mathcal{C}_{n,m(\lambda,n)}, \quad \widehat{C}_1 \xleftarrow{\$} \mathcal{E}_{C_1\|R,m'_1}, \quad \widehat{C} \xleftarrow{\$} \mathcal{E}_{C_1\|C_2,m'_1+m'_2}, \quad \widehat{C}_2 \xleftarrow{\$} \mathcal{E}_{R^{-1}\|C_2,m'_2}, \quad C^* \leftarrow \begin{cases} \widehat{C}_1\|\widehat{C}_2, & \text{if } \beta = 0; \\ \widehat{C}, & \text{if } \beta = 1; \end{cases}$$

  and send C^* to $\mathcal{A}$.

- **Guess.** $\mathcal{A}$ outputs a bit β', which is the output of the experiment.

Informally, Assumption 1 (SCP) states that *concatenation* and *perfect* rc$i\mathcal{O}$ commute given sufficient *randomization at boundary* (m, R) and *expansion* ($m^{\#}$).

We note that the distributions in SCP are not efficiently sampleable. Some of our reductions need SCP samples, which is one (not the only) reason that the results in this section require primitives secure against non-uniform adversaries.

rc$i\mathcal{O}$ from SCP and Obfuscation of Random Circuits. The work of [CCMR24] constructs an rc$i\mathcal{O}$ based on Assumption 1 and obfuscators for random circuits. They also propose candidate obfuscators for random circuits. We refer the readers to [CCMR24] for the details, and abstract their result as follows.

Lemma 15 (rc$i\mathcal{O}$ [CCMR24]). *There exists an* rc$i\mathcal{O}$ *scheme*[19] *(Definition 16) based on Assumption 1 and obfuscation of random circuits of $m^{\#}$ gates, where $m^{\#}$ is the parameter in Assumption 1.*

Remark 7 (hardened Toffoli). The work of [CCMR24] proves that rc$i\mathcal{O}$ implies $i\mathcal{O}$ via a *hardened* Toffoli construction, which embeds any normal circuit C into a reversible one R with $R(x, y, z) = (x, y \oplus C(x), z)$ whenever $z = 0^\star$ — except that the hardened version guards against certain garbage values in (ancillae) z so that equivalent C's always yield equivalent R's. In this work, we avoid hardened Toffoli to maximize the efficiency of our instantiations.

8.2 Permutable PRP

Assumption 1 (SCP) essentially allows us to permute a random circuit (wide enough) by any fixed circuit, provided with sufficient expansion. The distributions in the SCP assumption are not efficiently sampleable, but it can be swept under a non-uniformly secure rc$i\mathcal{O}$.

Ingredients of Construction 7. Let

- rc$i\mathcal{O}$ be a reversible-circuit obfuscator, and
- m, $m^{\#}$ polynomials in Assumption 1 (SCP).

[19] We note that their proof uses non-uniform reductions, so their result is a non-uniformly secure rc$i\mathcal{O}$ from non-uniform SCP and non-uniform security of the underlying random-circuit obfuscator.

Construction 7 (PPRP from rc$i\mathcal{O}$ and SCP). Define (see Definition 2)

$$G := \{G_N\}_{N\in\mathbb{N}}, \qquad G_N := \begin{cases} \mathcal{C}_{N,\star}, & \text{if } N \geq 4 \text{ and } N \geq \lambda; \\ \{1\}, & \text{otherwise.} \end{cases}$$

Here, "1" is the identity. For $N < \max\{4, \lambda\}$, the scheme does something uninteresting (e.g., $\pi = \pi_{1,0} = \pi_{1,1}$ are always the identity). Our IP-PRP PPRP for G works as follows for $N \geq \max\{4, \lambda\}$.

- $\mathsf{Gen}(1^N)$ samples and outputs $\pi \xleftarrow{\$} \mathcal{C}_{N,m(N)}$.
- $\mathsf{Permute}(\pi, \Gamma, b)$ parses N from π (it has N wires and $|\pi| = m(N)$ gates). The algorithm pads either π (when $b = 0$) or $\Gamma\|\pi$ (when $b = 1$) to $(|\Gamma| + |\pi|)m^{\#}(N)$ gates to obtain π'. It runs and outputs $\pi_{\Gamma,b} \xleftarrow{\$} \text{rc}i\mathcal{O}(\pi')$.
- $\mathsf{Eval}(\pi, x)$ evaluates and outputs $y \leftarrow \pi(x)$.
- $\mathsf{Eval}^{-1}(\pi, y)$ evaluates and outputs $x \leftarrow \pi^{-1}(y)$. (Note that $\pi \in \mathcal{C}_{N,\star}$, permuted or not, so π^{-1} can be evaluated easily.)

The correctness of Construction 7 follows directly from that of rc$i\mathcal{O}$.

Theorem 16 (¶). *Suppose* rc$i\mathcal{O}$ *is non-uniformly secure (Definition 16) and Assumption 1 holds (with respect to m, $m^{\#}$), then Construction 7 is a secure IP-PRP for G (Definition 2).*

The proof uses time-preserving reductions. The reduction from breaking rc$i\mathcal{O}$ *security to breaking PPRP security requires extra advice of size related to m, $m^{\#}$. The advantage relation is $\varepsilon_{\mathsf{PPRP}} = 2\varepsilon_{\text{rc}i\mathcal{O}} + 2\varepsilon_{\text{scp}}$.*

Proof (Theorem 16). Let "1" be a fixed identity gate and denote by π_* the permuted key given to the adversary. Consider the following hybrids.

- H_0^β is $\mathsf{Exp}_{\text{pprp}}^\beta$, where

$$\pi' \leftarrow \begin{cases} \pi\|1^{(|\Gamma|+m(N))m^{\#}(N)-m(N)}, & \text{if } \beta = 0; \\ \Gamma\|\pi\|1^{(|\Gamma|+m(N))m^{\#}(N)-|\Gamma|-m(N)}, & \text{if } \beta = 1; \end{cases} \qquad \pi_* \xleftarrow{\$} \text{rc}i\mathcal{O}(\pi').$$

  Recall that $|\pi| = m(N)$ and $|\pi'| = (|\Gamma| + m(N))m^{\#}(N)$ in both cases.

- In H_1^β, we replace the circuit given to rc$i\mathcal{O}$ by a random equivalent one:

$$C_1 \leftarrow \begin{cases} 1^{|\Gamma|}, & \text{if } \beta = 0; \\ \Gamma, & \text{if } \beta = 1; \end{cases} \qquad C_2 \leftarrow \begin{cases} 1^{|\Gamma|}, & \text{if } \beta = 0; \\ \Gamma^{-1}, & \text{if } \beta = 1; \end{cases}$$
$$\widehat{C}_1 \xleftarrow{\$} \mathcal{E}_{C_1\|\pi,\,(|\Gamma|+m(N))m^{\#}(N)}, \qquad \widehat{C}_2 \xleftarrow{\$} \mathcal{E}_{\pi^{-1}\|C_2,\,(m(N)+|\Gamma|)m^{\#}(N)},$$
$$\pi_* \xleftarrow{\$} \text{rc}i\mathcal{O}(\boxed{\widehat{C}_1}).$$

  By definition, $\widehat{C}_1$ and π' have the same size and functionality. We have $\mathsf{H}_0^\beta \approx \mathsf{H}_1^\beta$ by rc$i\mathcal{O}$ security.

  This relies on non-uniform security since $\mathcal{E}$ is not efficiently sampleable.

- In H_2^β, we change how $\widehat{C}_1$ is generated:

$$C_1 \leftarrow \begin{cases} 1^{|\Gamma|}, & \text{if } \beta = 0; \\ \Gamma, & \text{if } \beta = 1; \end{cases} \qquad C_2 \leftarrow \begin{cases} 1^{|\Gamma|}, & \text{if } \beta = 0; \\ \Gamma^{-1}, & \text{if } \beta = 1; \end{cases}$$

$$\boxed{\widehat{C}_1 \| \widehat{C}_2} \xleftarrow{\$} \mathcal{E}_{C_1\|C_2,\, 2(|\Gamma|+m(N))m^{\#}(N)} \qquad \text{with } |\widehat{C}_1| = (|\Gamma| + m(N))m^{\#}(N),$$
$$\pi_* \xleftarrow{\$} \mathrm{rc}i\mathcal{O}(\widehat{C}_1).$$

We have $\mathsf{H}_1^\beta \approx \mathsf{H}_2^\beta$ by Assumption 1. (R from the assumption becomes our π.)

Observe that $\mathsf{H}_2^0 \equiv \mathsf{H}_2^1$ as $\mathcal{E}_{C,m'}$ depends only on the permutation computed by C and in both cases, $C_1 \| C_2$ computes the identity.

The proof concludes by hybrid argument. □

8.3 Reversible-Circuit Trigger

Lemma 2 (random triggers) is a central tool in building public-key primitives from PPRP and $i\mathcal{O}$, and a reversible-circuit variant is needed here. The functionality of such a trigger potentially activated at $r \in \{0,1\}^N$ is

$$\mathcal{O}_N^{\mathrm{rc}}(x,y,z) = (x,y,z), \qquad \delta_r^{\mathrm{rc}}(x,y,z) = \begin{cases} (x, y \oplus 1, z), & \text{if } x = r; \\ (x,y,z), & \text{if } x \neq r; \end{cases} \qquad \begin{array}{l} \text{for } x \in \{0,1\}^N, \\ y,z \in \{0,1\}. \end{array}$$

We implement them and prove their security. Our scheme avoids the overhead of instantiating Section A using injective OWF and rc$i\mathcal{O}$ with hardened Toffoli (Remark 7), notably in the number of ancillae. We need exactly one ancilla (z), which is necessary to make δ_r^{rc} an even permutation.

Lemma 17 (reversible-circuit random triggers; ¶). *Based on non-uniformly secure* rc$i\mathcal{O}$ *and* UniqueSAT $\notin$ i.o.-PromiseP/poly,[20] *one can construct a* reversible-circuit trigger scheme*, with syntax and correctness modified from Lemma 2 (security is unmodified) as follows. (See Figure 2.)*

- $\not\Delta^{\mathrm{rc}}(1^\lambda, 1^N)$ *outputs a circuit in* $\mathcal{C}_{N+2,\star}$ *implementing* $\mathcal{O}_N^{\mathrm{rc}}$ *when* $N \geq \lambda \geq 1$.
- $\Delta^{\mathrm{rc}}(1^\lambda, r)$ *outputs a circuit in* $\mathcal{C}_{|r|+2,\star}$ *implementing* δ_r^{rc} *when* $|r| \geq \lambda \geq 1$.

It has non-uniform, (only!) polynomial security.

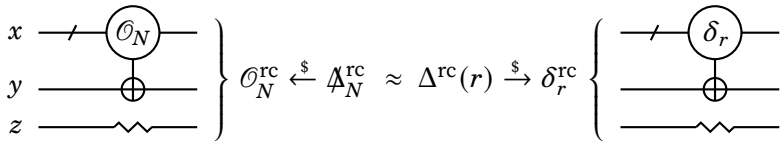


Figure 2. Reversible-circuit triggers. Wavy line indicates use as ancilla. ($\mathcal{O}_N$-controlled NOT is the identity, depicted for conceptual reasons.)

Remark 8 (security in Lemma 17). We always require the ingredient rc$i\mathcal{O}$ in Lemma 17 to be *non-uniformly* secure (satisfied by Lemma 15), regardless of the desired (non-) uniformity of trigger scheme security, due to the use of worst-case hardness. The reduction needs the hard UniqueSAT (non-)instances as (extra) advice.

[20] This can be further simplified to SAT $\notin$ i.o.-P/poly due to [VV85].

The uniformity of our trigger inherits from the worst-case hardness assumption. If we relax the latter to UniqueSAT $\notin$ i.o.-PromiseBPP, the trigger scheme becomes only uniformly (polynomially) secure. To obtain triggers with non-uniform subexponential security (both time and advantage), we need (non-uniformly) subexponentially secure rc$i\mathcal{O}$ and UniqueSAT $\notin$ i.o.-PromiseSIZE$(2^{|C|^c})$ for some $c > 0$. (But we only instantiate Lemma 7 with reversible circuits, so we are content with polynomial security.) For both cases, UniqueSAT hardness can also be replaced by SAT hardness due to [VV85].

Reversibly Evaluating 3-CNF. Our proof for Lemma 17 is to obfuscate a reversible circuit that tests a UniqueSAT (non-)instance (converted to 3-CNF) with the input as its assignment. The following claim comes handy.

Claim 18 (¶). *There exists an efficient deterministic algorithm that on input a 3-CNF*

$$f : \{0,1\}^N \to \{0,1\}, \qquad f(x) = \bigwedge_{i\in[M]} \psi_i(x),$$
$$(i \in [M]) \quad \psi_i(x) = L_{i,1} \vee L_{i,2} \vee L_{i,3}, \qquad (j \in [3]) \quad L_{i,j} \in \left\{ x_k, \neg x_k \,\middle|\, k \in [N] \right\},$$

outputs $C \in \mathcal{C}_{N+2,\star}$ with $|C| = \mathrm{O}(\max\{N, M^2/N\})$ that computes $(x, y, z) \mapsto (x, y \oplus f(x), z)$ for $x \in \{0,1\}^N$ and $y, z \in \{0,1\}$.

Proof (Claim 18). We construct an algorithm $A(N, F, \mathfrak{y}, \mathfrak{z})$, where $F = \wedge_{i\in[M]} \Psi_i$ is a 3-CNF over N variables (wires in $[N+2]$) and $\mathfrak{y}, \mathfrak{z} \in [N+2]$ are distinct wires not read by Ψ_i's. Here, $\mathfrak{y}$ is the output and $\mathfrak{z}$ is the ancilla. Given such input, A outputs a reversible circuit $C \in \mathcal{C}_{N+2,\star}$ such that

$$C(X_1 \| \cdots \| X_{N+2}) = X'_1 \| \cdots \| X'_{N+2} \qquad \text{with } X'_k = \begin{cases} X_k \oplus F(X_1 \| \cdots \| X_{N+2}), & \text{if } k = \mathfrak{y}; \\ X_k, & \text{if } k \neq \mathfrak{y}. \end{cases}$$

The task at hand can be completed as $A(N, f, N+1, N+2)$.

We use the idea of constructing C^{N}NOT from CNOT and C^{2}NOT in [BBC+95], except that we need to compute the conjunction of M *clauses* instead of N *variables*. If $N \leq 6$, the algorithm A simply outputs the circuit guaranteed by Lemma 14. Otherwise, it recursively builds the circuit according to M as follows.

- ($M \leq 2$.) Use Lemma 14 to obtain a circuit C on $\mathfrak{y}$, $\mathfrak{z}$, and the wires read by Ψ_i's. This involves up to $3M + 2 \leq 8$ wires, so brute force is fine and the number of gates is $|C| = \mathrm{O}(1)$.
- ($2 < M \leq N/4$.) Let $1 \leq \mathfrak{y}_1 < \cdots < \mathfrak{y}_{M-1} \leq N+2$ be $(M-1)$ distinct wires of the smallest indices that are neither $\mathfrak{y}$ or $\mathfrak{z}$ nor read by Ψ_i's. (Such wires exist since there are at least $N - 3M \geq M$ unused wires.) Set $\mathfrak{y}_M \leftarrow \mathfrak{y}$, compute

$$C'_1 \leftarrow A(N, \Psi_1, \mathfrak{y}_1, \mathfrak{z}), \qquad C_1 \leftarrow C'_1,$$
$$(i \in [2, M]) \quad C'_i \leftarrow A(N, \Psi_i \wedge (X_{\mathfrak{y}_{i-1}} \vee X_{\mathfrak{y}_{i-1}} \vee X_{\mathfrak{y}_{i-1}}), \mathfrak{y}_i, \mathfrak{z}), \quad C_i \leftarrow C'_i \| C_{i-1} \| C'_i,$$

  and output $C \leftarrow C_M \| C_{M-1}$. See Figure 3 for a depiction.

  Correctness follows from an induction on $i \in [M]$ that the effect of C_i is precisely $X_{\mathfrak{y}_{i'}} \leftarrow X_{\mathfrak{y}_{i'}} \oplus \wedge_{i'' \leq i'} \Psi_{i''}$ for all $i' \leq i$.

  Each C'_i is constant-size by the case of $M \leq 2$, and C is the concatenation of $4M - 4 < N$ such circuits. Therefore, the number of gates is $|C| = \mathrm{O}(N)$.

- ($M > \max\{2, N/4\}$.) Find the lexicographically smallest $(\mathfrak{w}, S)$ such that $X_{\mathfrak{w}}$ is not read by $\{\Psi_i\}_{i\in S}$ where $S \subseteq [M]$ and $|S| = \lfloor M/2 \rfloor$. To do so, tally the number M_k of clauses reading X_k for all $k \in [N+2]$, and pick

$$\mathfrak{w} = \min\Big\{ k \in [N+2] \;\Big|\; k \neq \mathfrak{y}, \mathfrak{z} \text{ and } M_k \le M/2 \Big\}.$$

Such $\mathfrak{w}$ exists because $\frac{1}{N}\sum_{k\neq\mathfrak{y},\mathfrak{z}} M_k \le 3M/N \le M/2$ (recall that $N \ge 7$ in this recursion). Now that at least $M/2$ clauses do not read $X_{\mathfrak{w}}$, we pick S to be the smallest $\lfloor M/2 \rfloor$ indices of them. Split F and recursively compute

$$F_1 \leftarrow \bigwedge_{i\in S} \Psi_i, \qquad C_1 \leftarrow A(N, F_1 \wedge (X_{\mathfrak{z}} \vee X_{\mathfrak{z}} \vee X_{\mathfrak{z}}), \mathfrak{y}, \mathfrak{w}),$$
$$F_2 \leftarrow \bigwedge_{i\in[M]\setminus S} \Psi_i, \qquad C_2 \leftarrow A(N, F_2, \mathfrak{z}, \mathfrak{y}),$$

and output $C \leftarrow C_1 \| C_2 \| C_1 \| C_2$. See Figure 4 for correctness.

The number of gates is $|C| = \mathrm{O}(4^{\log_2(M/(N/4))} \cdot N) = \mathrm{O}(M^2/N)$ — at each recursion, M is reduced by half and the output quadruples, until $M \le N/4$ (the previous case). □

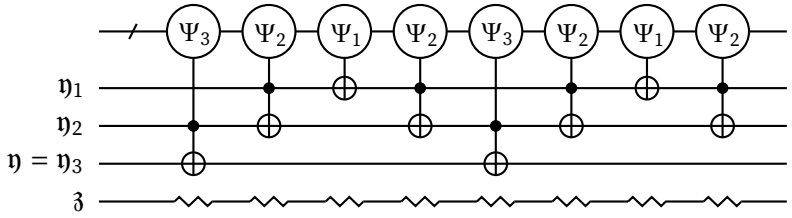


Figure 3. Converting 3-CNF to reversible circuits ($2 < M \le N/4$, e.g., $M = 3$).

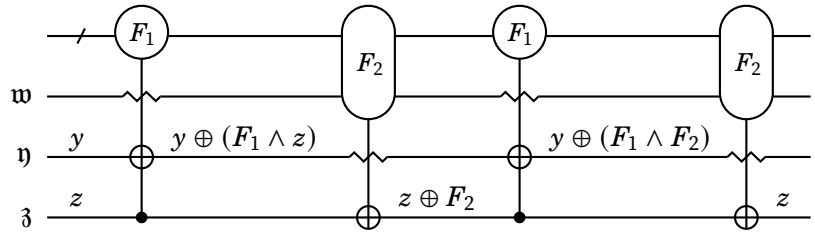


Figure 4. Converting 3-CNF to reversible circuits ($M > \max\{2, N/4\}$).

Proving Lemma 17. We use the observation from [IL25] that the accepting input of uniquely satisfiable circuits can be rerandomized. Given $C : \{0,1\}^n \to \{0,1\}$, one can rerandomize it into $C_r(x) := C(x \oplus r)$ with $r \xleftarrow{\$} \{0,1\}^n$: if $C \in \mathsf{UniqueSAT}$, then C_r implements a random point function; otherwise, $C \in \mathsf{UNSAT}$ and C_r implements the all-zero function. An adversary distinguishing deactivated and random triggers (obfuscated all-zero or random point functions) can also distinguish obfuscated C_r's for $C \in \mathsf{UNSAT}$ and $C \in \mathsf{UniqueSAT}$, which solves $\mathsf{UniqueSAT}$.

Proof (Lemma 17). Let $\mathrm{rc}i\mathcal{O}$ be a reversible-circuit $i\mathcal{O}$. The trigger scheme works as follows.

- $\not\Delta^{\mathrm{rc}}(1^N)$ outputs $\mathrm{rc}i\mathcal{O}(\mathcal{O}_N^{\mathrm{rc}})$ if $N \geq \lambda \geq 1$.
- $\Delta^{\mathrm{rc}}(r)$ outputs $\mathrm{rc}i\mathcal{O}(\delta_r^{\mathrm{rc}})$ if $|r| \geq \lambda \geq 1$.

Here, $\mathcal{O}_N^{\mathrm{rc}}$ is implemented by an appropriate number of identity gates. As for δ_r^{rc}, its cycle decomposition is $(r\|0\|0 \quad r\|1\|0)(r\|0\|1 \quad r\|1\|1)$, so an $\mathrm{O}(|r|)$-size circuit for it can be obtained by Lemma 14, then padded for obfuscation. Correctness is readily verified.

We prove its security by contradiction. Suppose for some $k \in \mathbb{N}$, infinite set $\mathcal{I} \subseteq \mathbb{N}$, and non-uniform polynomial-time $\mathcal{A}$, it could distinguish $\mathsf{Exp}^0_{\mathrm{trigger}}$ and $\mathsf{Exp}^1_{\mathrm{trigger}}$ with positive advantage at least $6\lambda^{-k}$ for all $\lambda \in \mathcal{I}$. Define

$$\varepsilon_\beta := \Pr[\mathsf{Exp}^\beta_{\mathrm{trigger}}(\mathcal{A}) = 1], \qquad E_0 := \lceil (\varepsilon_0 + \lambda^{-k})\lambda^k \rceil, \qquad E_1 := \lfloor (\varepsilon_1 - \lambda^{-k})\lambda^k \rfloor,$$

then $\varepsilon_1 - \varepsilon_0 \geq 6\lambda^{-k}$ hence $E_1 - E_0 \geq 2$ for all $\lambda \in \mathcal{I}$.

Given any C for which $C \in \mathsf{UNSAT}$ or $C \in \mathsf{UniqueSAT}$ is to be decided, let $\mathcal{B}_1(C)$ do the following.

1. Let λ be the bit-length of C. Launch $\mathcal{A}(1^\lambda)$ and receive 1^N from it. Abort and output 0 if $N < \lambda$ or $\lambda < 1$.
2. Convert C to 3-CNF f with exactly N variables and $M \leq 6N$ clauses, with the number of satisfying assignments preserved.

   Recall that the standard reduction [Kar72] is to create a variable in f for every gate in C (the number of which cannot exceed $\lambda \leq N$) and make f verify correct evaluation and acceptance of C (in no more than $4\lambda + 1 \leq 5N$ clauses). We simply add additional variables x_i to f and constrain them (say) to be true by joining $(x_i \vee x_i \vee x_i)$ to the clauses. This yields the desired 3-CNF.
3. Convert f to a reversible circuit $C' \in \mathcal{C}_{N+2,\star}$ using Claim 18, then C' computes $(x, y, z) \mapsto (x, y \oplus f(x), z)$ for $x \in \{0,1\}^N$ and $y, z \in \{0,1\}$ and has $\mathrm{O}(N)$ gates.
4. Sample $r \xleftarrow{\$} \{0,1\}^N$ and pad

   $$\mathsf{NOT}[r\|0\|0] \quad \| \quad C' \quad \| \quad \mathsf{NOT}[r\|0\|0]$$

   with identity gates into C''.

   Here, $\mathsf{NOT}[r'] \in \mathcal{C}_{|r'|,\star}$ consists of NOT gates on wires $\{\, i \in [|r'|] \,|\, r'_i = 1 \,\}$. Its effect is to XOR the wire values by r'.
5. Run and send $\mathrm{rc}i\mathcal{O}(C'')$ to $\mathcal{A}$. Output the bit produced by $\mathcal{A}$.

If $C \in \mathsf{UNSAT}$, then C'' is equivalent to $\mathcal{O}_N^{\mathrm{rc}}$. If C is uniquely satisfied by w_1^*, then f is uniquely satisfied by w_2^* (efficiently computable from C, w_1^*), and C'' is equivalent to $\delta^{\mathrm{rc}}_{w_2^* \oplus r}$. By the non-uniform security of $\mathrm{rc}i\mathcal{O}$, when C is long enough, we have

$$\Pr[\mathcal{B}_1(C) = 1] \begin{cases} \leq \varepsilon_0 + \lambda^{-k} \leq E_0/\lambda^k, & \text{if } C \in \mathsf{UNSAT}; \\ \geq \varepsilon_1 - \lambda^{-k} \geq E_1/\lambda^k, & \text{if } C \in \mathsf{UniqueSAT}; \\ (\text{irrelevant}), & \text{otherwise.} \end{cases}$$

Roughly speaking, to bound the maximum difference between $\Pr[\mathcal{B}_1(C) = 1]$ and ε_0 or ε_1, the reduction (against rc$i\mathcal{O}$ security) uses either $C \in \mathsf{UNSAT}$ or $C \in \mathsf{UniqueSAT}$ and its w_1^* as extra advice.

We apply standard amplification. Let $\mathcal{B}_2(C)$ receive $E_* = E_0 + E_1$ as advice. It sets λ to the bit-length of C, runs $\mathcal{B}_1(C)$ with fresh randomness for λ^{2k+1} times, and counts the number E of 1's produced by $\mathcal{B}_1$. The algorithm outputs 1 if $E/\lambda^{2k+1} \geq E_*/(2\lambda^k)$, and 0 otherwise. Chernoff bound shows that $\mathcal{B}_2$ errs with probability strictly less than $2^{-\lambda}$ when λ is large enough and falls in $\mathcal{I}$ (reliant on $E_1 - E_2 \geq 2$). The best randomness derandomizes $\mathcal{B}_2$, implying $\mathsf{UniqueSAT} \in \mathsf{i.o.\text{-}PromiseP/poly}$. $(\rightarrow\!\!\!\!\!\not\in)$

We now analyze the required amount of padding. Recall that C' (before rerandomization) has $O(N)$ gates and $\mathsf{NOT}[r\|0\|0]$ has up to N gates. Therefore, it suffices to pad all the circuits to some known $\Theta(N)$ gates. The hidden constants can be fully worked out by recursively tracing the proof of Claim 18.

Lastly, to base uniform security on $\mathsf{UniqueSAT} \notin \mathsf{i.o.\text{-}PromiseBPP}$, we can make $\mathcal{B}_2$ estimate E_0, E_1 to exponential confidence instead of advising it with E_*. □

8.4 Useful Gadgets of Reversible Circuits

We present some gadgets of reversible circuits that will be useful for proving the security of our KEM and PKE instantiations.

Simple, Controlled Partial, and Self-Controlled Toffoli. Let $C \in \mathcal{C}_{n,\star}$ be a reversible circuit, define $\mathsf{TOF}[C]$ to be a $2n$-wire reversible circuit as follows: apply C on the first n wires; XOR (using CNOT) each of the first n wires into the corresponding one among the last n wires; lastly, apply C^{-1} on the first n wires. As depicted in Figure 5, it computes

$$\mathsf{TOF}[C](x, y) = (x, y \oplus C(x)) \qquad \text{for } x, y \in \{0,1\}^n.$$

In the proofs, we need to apply $\mathsf{TOF}[C]$ conditionally, where the condition is about x. We achieve this in two steps.

Let $X, \{k\}, Y' \subseteq [2n]$ be pairwise disjoint lists of distinct wires and $X' \subseteq X$ such that $|X| = n$ and $|X'| = |Y'|$. We define $\mathsf{CPTOF}[C, X : X' \xrightarrow{k} Y']$, the *controlled partial* Toffoli, as the circuit in Figure 6, which involves C, a bunch of $\mathsf{C^2NOT}$'s, and C^{-1}. If X (resp. X', Y') is $\{\mathfrak{x}_i\}_i$ (resp. $\{\mathfrak{x}'_i\}_i$, $\{\mathfrak{y}'_i\}_i$), then

$$\mathsf{CPTOF}[C, X : X' \xrightarrow{k} Y'](z_1\|\cdots\|z_{2n}) = z'_1\|\cdots\|z'_{2n}, \quad z'_i = \begin{cases} z_i, & \text{if } i \notin Y'; \\ z_i \oplus (z_k \wedge w_{\mathfrak{x}'_i}), & \text{if } i = \mathfrak{y}'_i; \end{cases}$$

where $w_{\mathfrak{x}_1}\|\cdots\|w_{\mathfrak{x}_n} = C(z_{\mathfrak{x}_1}\|\cdots\|z_{\mathfrak{x}_n})$, i.e., after C over X, wires X' are XOR'd into Y' if and only if wire k is one, and wires X are restored.

Assume $n \geq 2$. Let $\mathcal{T} : \{0,1\}^n \to \{0,1\}$ be a function and suppose $\mathcal{T}^{\mathrm{rc}}$ is a reversible circuit such that $\mathcal{T}^{\mathrm{rc}}(x, y, z) = (x, y \oplus \mathcal{T}(x), z)$ for all $x \in \{0,1\}^n$ and $y, z \in \{0,1\}$. (This notation is consistent with triggers.) We define $\mathsf{SCTOF}[\mathcal{T}^{\mathrm{rc}}, C]$, the *self-controlled* Toffoli, as the circuit in Figure 7, i.e.,

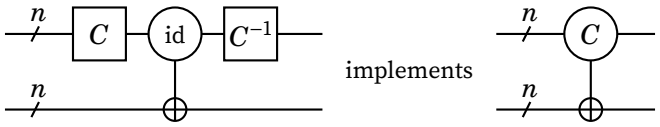


Figure 5. The circuit $\mathsf{TOF}[C]$.

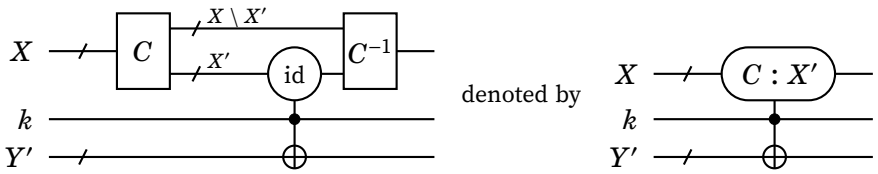


Figure 6. The circuit $\mathsf{CPTOF}[C, X : X' \xrightarrow{k} Y']$.

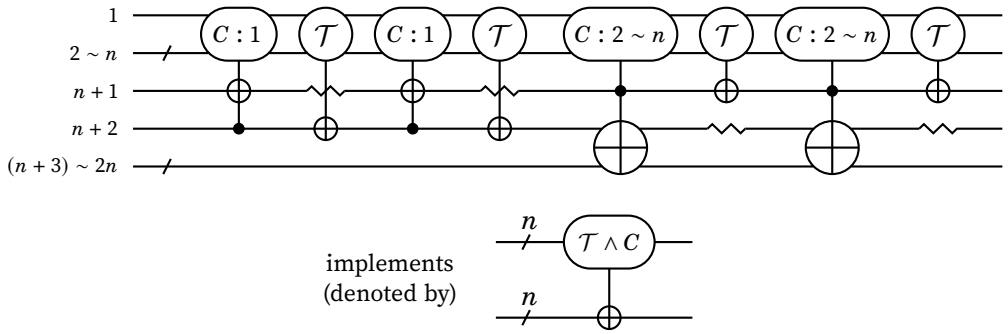


Figure 7. The circuit $\mathsf{SCTOF}[\mathcal{T}^{\mathrm{rc}}, C]$.

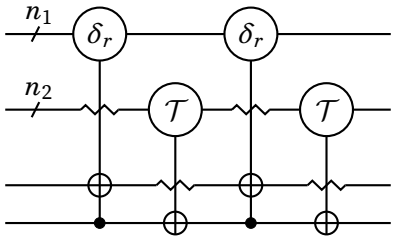


Figure 8. The circuit $r \circledcirc \mathcal{T}^{\mathrm{rc}}$.

$$\begin{aligned}\mathsf{SCTOF}[\mathcal{T}^{\mathrm{rc}}, C] := \mathsf{CPTOF}[C, 1 \sim n : \quad & 1 \xrightarrow{n+2} n+1 \quad] \| \mathcal{T}' \\ \| \mathsf{CPTOF}[C, 1 \sim n : \quad & 1 \xrightarrow{n+2} n+1 \quad] \| \mathcal{T}' \\ \| \mathsf{CPTOF}[C, 1 \sim n : \quad & 2 \sim n \xrightarrow{n+1} (n+2) \sim 2n \ ] \| \mathcal{T}'' \\ \| \mathsf{CPTOF}[C, 1 \sim n : \quad & 2 \sim n \xrightarrow{n+1} (n+2) \sim 2n \ ] \| \mathcal{T}'',\end{aligned}$$

where $\mathcal{T}'$ (resp. $\mathcal{T}''$) is $\mathcal{T}^{\mathrm{rc}}$ rewired to use wires $(n+2)$, $(n+1)$ (resp. in reverse order) as the target and the ancilla. It computes

$$\mathsf{SCTOF}[\mathcal{T}^{\mathrm{rc}}, C](x, y) = \begin{cases} (x, y \oplus C(x)), & \text{if } \mathcal{T}(x) = 1; \\ (x, y), & \text{if } \mathcal{T}(x) = 0. \end{cases}$$

Trigger-Controlled NOT. Given $\mathcal{T}^{\mathrm{rc}}$ and any bit-string $y' \in \{0,1\}^n$ for $n \geq 2$, we can efficiently implement

$$\mathsf{C'NOT}[\mathcal{T}^{\mathrm{rc}}, y'](x, y) = \begin{cases} (x, y \oplus y'), & \text{if } \mathcal{T}(x) = 1; \\ (x, y), & \text{if } \mathcal{T}(x) = 0; \end{cases} \qquad \text{for } x, y \in \{0,1\}^n.$$

The naïve method is to duplicate $\mathcal{T}^{\mathrm{rc}}$ multiple times (the Hamming weight of y') and rewire the copies to act on the targets (selected by the ones in y'). The result is a circuit of size up to $|\mathcal{T}^{\mathrm{rc}}| \cdot n$.

A more efficient way is to use the trick of SCTOF. If $y'_i = 0$ for some $i \in [n]$, then do half of SCTOF, except that CPTOF is replaced by CNOT's (controlled by wire $(n+i)$ with the targets being $(n+j)$ for each $j \in [n]$ such that $y'_j = 1$). Otherwise, $y' = 1^n$, do the full SCTOF with CPTOF replaced by CNOT's, the first time potentially negating wire $(n+1)$ and the second time potentially negating wires $(n+2)$ to $2n$. This gives a circuit of size up to $4(|\mathcal{T}^{\mathrm{rc}}| + n)$.

Conjugated Triggers. Given $\mathcal{T}^{\mathrm{rc}}$ and $C \in \mathcal{C}_{n,\star}$, we define $\mathcal{T}^{\mathrm{rc}}\langle C \rangle$ to be the following $(n+2)$-wire circuit: first apply C on the first n wires; then apply $\mathcal{T}^{\mathrm{rc}}$; lastly, apply C^{-1} on the first n wires. It computes

$$\mathcal{T}^{\mathrm{rc}}\langle C \rangle(x, y, z) = (x, y \oplus \mathcal{T}(C(x)), z) \qquad \text{for } x \in \{0,1\}^n,\ y, z \in \{0,1\}.$$

Known-Trigger-Controlled Unknown Triggers. Let $r \in \{0,1\}^{n_1}$ and $\mathcal{T}^{\mathrm{rc}}$ be a reversible circuit such that

$$\mathcal{T}^{\mathrm{rc}}(x_2, y, z) = (x_2, y \oplus \mathcal{T}(x_2), z) \qquad \text{for } x_2 \in \{0,1\}^{n_2},\ y, z \in \{0,1\},$$

where $n_2 \geq 1$. We define $r \oslash \mathcal{T}^{\mathrm{rc}}$ to be the circuit in Figure 8. Note that the first/third blocks are nothing but $\delta^{\mathrm{rc}}_{r\|1}$ rewired. It computes

$$(r \oslash \mathcal{T}^{\mathrm{rc}})(x_1, x_2, y, z) = (x_1, x_2, y \oplus (\delta_r(x_1) \wedge \mathcal{T}(x_2)), z)$$

for $x_1 \in \{0,1\}^{n_1}$, $x_2 \in \{0,1\}^{n_2}$, $y, z \in \{0,1\}$.

8.5 KEM and (Plaintext-Selective) PKE with CCA Security

We present the reversible-circuit instantiation of CCA-secure KEM (Construction 2) and plaintext-selective CCA-secure PKE. Our instantiation uses reversible-circuit PPRP (Construction 7) and rc$i\mathcal{O}$ in place of $i\mathcal{O}$. To capture both KEM and PKE, we prove a reversible-circuit version of Lemma 7.

As explained in Remark 6, if π is a reversible-circuit PPRP, then rc$i\mathcal{O}(\pi)$ does *not* prevent evaluation of π^{-1}. The simplest idea to fix this is to obfuscate its (simple) Toffoli version. The proof does not use additional ancillae, avoiding the costly hardened Toffoli transformation (Remark 7).

A fully adaptive CCA-secure PKE can be obtained by introducing ancillae (akin to hardened Toffoli transformation). We are currently unable to achieve it *without ancillae*. Adaptivity is discussed at the end (Remark 10).

Ingredients of Constructions 8 and 9. Let

- $\mathsf{PPRP} = (\mathsf{PPRP.Gen}, \mathsf{PPRP.Permute}, \mathsf{PPRP.Eval}, \mathsf{PPRP.Eval}^{-1})$ be an IP-PRP such that a key (permuted or not) for input length N is always some $\pi \in \mathcal{C}_{N,\star}$ and $\mathsf{PPRP.Eval}$ is simply evaluating π (e.g., Construction 7), and
- rc$i\mathcal{O}$ a reversible-circuit obfuscator.

Construction 8 (reversible-circuit KEM, cf. Construction 2). It works as follows.

- $\mathsf{Gen}()$ runs
$$\pi \xleftarrow{\$} \mathsf{PPRP.Gen}(1^\lambda), \qquad \widetilde{\pi} \xleftarrow{\$} \text{rc}i\mathcal{O}(\mathsf{TOF}[\pi]),$$
$$\sigma \xleftarrow{\$} \mathsf{PPRP.Gen}(1^\lambda), \qquad \widetilde{\sigma} \xleftarrow{\$} \text{rc}i\mathcal{O}(\mathsf{TOF}[\sigma]),$$
sets $\mathsf{pk} \leftarrow (\widetilde{\pi}, \widetilde{\sigma})$ and $\mathsf{sk} \leftarrow (\pi^{-1}, \sigma)$, and outputs $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Enc}(\mathsf{pk})$ first parses pk into $(\widetilde{\pi}, \widetilde{\sigma})$. It then samples $r \xleftarrow{\$} \{0,1\}^\lambda$ and computes $r \| \mathsf{ct} \leftarrow \widetilde{\pi}(r \| 0^\lambda)$ and $r \| k \leftarrow \widetilde{\sigma}(r \| 0^\lambda)$. The algorithm outputs (ct, k).
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ parses sk into (π^{-1}, σ). It computes and outputs $k \leftarrow \sigma(\pi^{-1}(\mathsf{ct}))$.

Construction 9 (reversible-circuit PKE, cf. Construction 1). It works as follows.

- $\mathsf{Gen}(1^n)$ samples $\pi \xleftarrow{\$} \mathsf{PPRP.Gen}(1^{n+\lambda})$ and obfuscates the simple Toffoli version of π into $\widetilde{\pi} \xleftarrow{\$} \text{rc}i\mathcal{O}(\mathsf{TOF}[\pi])$. It sets $\mathsf{pk} \leftarrow \widetilde{\pi}$, $\mathsf{sk} \leftarrow \pi^{-1}$, and outputs $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Enc}(\mathsf{pk}, x)$ parses pk into $\widetilde{\pi}$ and samples $r \xleftarrow{\$} \{0,1\}^\lambda$. The algorithm computes $x \| r \| \mathsf{ct} \leftarrow \widetilde{\pi}(x \| r \| 0^{n+\lambda})$ and outputs ct.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ parses sk into π^{-1}, runs $x \| r \leftarrow \pi^{-1}(\mathsf{ct})$, and outputs x (discarding the λ-bit r).

Both schemes are clearly correct.

Lemma 19 (Lemma 7 using reversible circuits; ¶). *Suppose* rc$i\mathcal{O}$ *and* PPRP *are secure (Definitions 16 and 2),* PPRP *supports all 3-cycles, and the conclusion of Lemma 17 (reversible-circuit triggers) holds, then* $\mathsf{Exp}^0_{\text{cca-rr}} \approx \mathsf{Exp}^1_{\text{cca-rr}}$*, where* $\mathsf{Exp}^\beta_{\text{cca-rr}}(\mathcal{A})$ *proceeds as follows.*

- **Challenge.** Launch $\mathcal{A}()$ and receive $v^* \in \{0,1\}^\star$ from it. Run
$$\pi \xleftarrow{\$} \mathsf{Gen}(1^{|v^*|+\lambda}), \qquad \sigma \xleftarrow{\$} \mathsf{Gen}(1^\lambda), \qquad r^* \xleftarrow{\$} \{0,1\}^\lambda,$$
$$y_0 \leftarrow \pi(v^* \| r^*), \qquad y_1 \xleftarrow{\$} \{0,1\}^{|v^*|+\lambda}, \qquad y^* \leftarrow y_\beta,$$

$$s_0 \leftarrow \sigma(r^*), \qquad s_1 \xleftarrow{\$} \{0,1\}^\lambda, \qquad s^* \leftarrow s_\beta,$$
$$\widetilde{P} \xleftarrow{\$} \mathrm{rc}i\mathcal{O}(\mathsf{TOF}[\pi]), \qquad \widetilde{Q} \xleftarrow{\$} \mathrm{rc}i\mathcal{O}(\mathsf{TOF}[\sigma]),$$
$$\widetilde{R} \xleftarrow{\$} \mathrm{rc}i\mathcal{O}(\mathsf{TOF}[\pi^{-1}] \| \mathsf{SCTOF}[\delta^{\mathrm{rc}}_{y^*}, \pi^{-1}]),$$

where the calls to $\mathrm{rc}i\mathcal{O}$ are appropriately padded. Send $(y^*, s^*, \widetilde{P}, \widetilde{Q}, \widetilde{R})$ to $\mathcal{A}$.

- **Guess.** $\mathcal{A}$ outputs a bit β', which is the output of the experiment.

The proof uses time-preserving reductions and the advantage relation is

$$|\varepsilon_{\text{cca-rr}}| \le 12 \cdot 2^{-\lambda} + 12|\varepsilon_{\mathrm{rc}i\mathcal{O}}| + 7|\varepsilon_{\text{trigger}}| + 2|\varepsilon_{\mathsf{PPRP}}|.$$

Corollary 20. *Suppose* $\mathrm{rc}i\mathcal{O}$ *and* PPRP *are secure (Definitions 16 and 2),* PPRP *supports all 3-cycles, and the conclusion of Lemma 17 (reversible-circuit triggers) holds, then Construction 8 is CCA-secure (Definition 6) and Construction 9 is plaintext-selective CCA-secure (Definition 5, except that* x^* *must be chosen prior to receiving* pk*).*

The proofs use time-preserving reductions and the advantage relations are $\varepsilon = \varepsilon_{\text{cca-rr}}$*, where* $\varepsilon_{\text{cca-rr}}$ *is defined in Lemma 19.*

Remark 9. The circuit $\widetilde{R}$ in Lemma 19 does $\mathsf{TOF}[\pi^{-1}]$ twice (cancelling each other) if $y = y^*$ and once otherwise, so it is a Toffoli version of

$$y \mapsto \begin{cases} 0^{|v^*|+\lambda}, & \text{if } y = y^*; \\ \pi^{-1}(y), & \text{otherwise;} \end{cases}$$

as opposed to mapping y^* to $\perp$ in Lemma 7 (normal-circuit version). This change does not affect its applications (Corollary 20) as the reductions never evaluate $\widetilde{R}(y^*)$ for the adversaries.

Proof (Lemma 19). It mimics that of Lemma 7, with the following modifications.

- π, σ cannot be permuted by transpositions (odd permutations). We instead permute them by 3-cycles.
- $\mathrm{rc}i\mathcal{O}$ is format-preserving for reversible circuits. We must implement all circuits in the hybrids using the same number of wires.

Sample $x' \xleftarrow{\$} \{0,1\}^{|v^*|+\lambda}$, $x'' \xleftarrow{\$} \{0,1\}^{|v^*|+\lambda}$, $r' \xleftarrow{\$} \{0,1\}^\lambda$, $r'' \xleftarrow{\$} \{0,1\}^\lambda$. Let $(\mathbb{\Delta}^{\mathrm{rc}}, \Delta^{\mathrm{rc}})$ be a trigger scheme and $\mathcal{T}^{\mathrm{rc}}_0$, $\mathcal{T}^{\mathrm{rc}}_{11}$, $\mathcal{T}^{\mathrm{rc}}_{12}$, $\mathcal{T}^{\mathrm{rc}}_{21}$, $\mathcal{T}^{\mathrm{rc}}_{22}$ triggers potentially activated at r^*, x', x'', r', r''.

Writing P, Q, R for the reversible circuits being obfuscated into $\widetilde{P}$, $\widetilde{Q}$, $\widetilde{R}$, we consider the following hybrids, which are fully specified by y^*, s^*, P, Q, R.

- G_0 is $\mathsf{Exp}^0_{\text{cca-rr}}$, where

$$y^* = \pi(v^* \| r^*), \qquad s^* = \sigma(r^*),$$
$$P = \mathsf{TOF}[\pi], \qquad Q = \mathsf{TOF}[\sigma], \qquad R = \mathsf{TOF}[\pi^{-1}] \| \mathsf{SCTOF}[\delta^{\mathrm{rc}}_{y^*}, \pi^{-1}].$$

- In G_1, we sample

$$\mathcal{T}^{\mathrm{rc}}_{11} \xleftarrow{\$} \mathbb{\Delta}^{\mathrm{rc}}_{|v^*|+\lambda}, \qquad \mathcal{T}^{\mathrm{rc}}_{12} \xleftarrow{\$} \mathbb{\Delta}^{\mathrm{rc}}_{|v^*|+\lambda}, \qquad \mathcal{T}^{\mathrm{rc}}_{21} \xleftarrow{\$} \mathbb{\Delta}^{\mathrm{rc}}_{\lambda}, \qquad \mathcal{T}^{\mathrm{rc}}_{22} \xleftarrow{\$} \mathbb{\Delta}^{\mathrm{rc}}_{\lambda}.$$

Additional logic is introduced into the circuits:

$$y^* = \pi(v^* \| r^*), \qquad s^* = \sigma(r^*),$$
$$P = \mathsf{TOF}[\pi] \boxed{\| \mathsf{SCTOF}[\mathcal{T}^{\mathrm{rc}}_{11}, \pi] \| \mathsf{C'NOT}[\mathcal{T}^{\mathrm{rc}}_{11}, y^*] \| \mathsf{SCTOF}[\mathcal{T}^{\mathrm{rc}}_{12}, \pi]},$$

$$Q = \mathsf{TOF}[\sigma] \boxed{\|\mathsf{SCTOF}[\mathcal{T}^{\mathrm{rc}}_{21}, \sigma]\|\mathsf{C'NOT}[\mathcal{T}^{\mathrm{rc}}_{21}, s^*]\|\mathsf{SCTOF}[\mathcal{T}^{\mathrm{rc}}_{22}, \sigma]},$$
$$R = \mathsf{TOF}[\pi^{-1}]\|\mathsf{SCTOF}[\delta^{\mathrm{rc}}_{y^*}, \pi^{-1}] \boxed{\|\mathsf{SCTOF}[\mathcal{T}^{\mathrm{rc}}_{11}\langle\pi^{-1}\rangle, \pi^{-1}]} \boxed{\|\mathsf{SCTOF}[\mathcal{T}^{\mathrm{rc}}_{12}\langle\pi^{-1}\rangle, \pi^{-1}]\|\mathsf{C'NOT}[\mathcal{T}^{\mathrm{rc}}_{12}\langle\pi^{-1}\rangle, v^*\|r^*]}.$$

Since all $\mathcal{T}^{\mathrm{rc}}$'s are $\mathcal{O}^{\mathrm{rc}}$'s, the newly added controlled operations are never done. By rc$i\mathcal{O}$ security, $\mathsf{G}_0 \approx \mathsf{G}_1$.

It is instructive to understand what the added operations do. For P: it first computes π unconditionally ("computes" in the sense of Toffoli); if $\mathcal{T}_{11}(v\|r) = 1$, it cancels π and computes y^*; if $\mathcal{T}_{12}(v\|r) = 1$, it also cancels π but does nothing more. (We assume that $\mathcal{T}_{11}$ and $\mathcal{T}_{12}$ do not collide.) In other words, P is the Toffoli version of

$$v\|r \mapsto \begin{cases} y^* = \pi(v^*\|r^*), & \text{if } \mathcal{T}_{11}(v\|r) = 1 \quad \text{(think } v\|r = x'\text{);} \\ 0^{|v^*|+\lambda}, & \text{if } \mathcal{T}_{12}(v\|r) = 1 \quad \text{(think } v\|r = x''\text{);} \\ \pi(v\|r), & \text{otherwise.} \end{cases}$$

Similarly, Q and R are Toffoli versions of

$$(Q) \quad r \mapsto \begin{cases} \sigma(r^*), & \text{if } \mathcal{T}_{21}(r) = 1; \\ 0^{\lambda}, & \text{if } \mathcal{T}_{22}(r) = 1; \\ \sigma(r), & \text{otherwise;} \end{cases} \qquad (R) \quad y \mapsto \begin{cases} 0^{|v^*|+\lambda}, & \text{if } \pi^{-1}(y) = v^*\|r^*; \\ 0^{|v^*|+\lambda}, & \text{if } \mathcal{T}_{11}(\pi^{-1}(y)) = 1; \\ v^*\|r^*, & \text{if } \mathcal{T}_{12}(\pi^{-1}(y)) = 1; \\ \pi^{-1}(y), & \text{otherwise.} \end{cases}$$

Notably, the branch leading to $v^*\|r^*$ in R will later be taken by x'', the preimage of $v^*\|r^*$ under the intended input permutation, $(v^*\|r^* \;\; x' \;\; x'')$.

- In G_2, we switch the triggers to

$$\mathcal{T}^{\mathrm{rc}}_{11} \xleftarrow{\$} \boxed{\Delta^{\mathrm{rc}}(x')}, \quad \mathcal{T}^{\mathrm{rc}}_{12} \xleftarrow{\$} \boxed{\Delta^{\mathrm{rc}}(x'')}, \quad \mathcal{T}^{\mathrm{rc}}_{21} \xleftarrow{\$} \boxed{\Delta^{\mathrm{rc}}(r')}, \quad \mathcal{T}^{\mathrm{rc}}_{22} \xleftarrow{\$} \boxed{\Delta^{\mathrm{rc}}(r'')}.$$

  By trigger scheme security, $\mathsf{G}_1 \approx \mathsf{G}_2$.

- In $\mathsf{G}_{2.5}$, we abort the experiment if $|\{v^*\|r^*, x', x''\}| < 3$ or $|\{r^*, r', r''\}| < 3$.

  We have $\mathsf{G}_2 \approx_{\mathrm{s}} \mathsf{G}_{2.5}$ at an error of no more than $6 \cdot 2^{-\lambda}$.

- In G_3, we sample

$$\pi_* \xleftarrow{\$} \mathsf{PPRP.Permute}(\pi, (v^*\|r^* \;\; x' \;\; x''), 0),$$
$$\sigma_* \xleftarrow{\$} \mathsf{PPRP.Permute}(\sigma, (r^* \;\; r' \;\; r''), 0),$$

  and rewrite the components with x', x'', r', r'', π_*, σ_* instead of the triggers and the non-permuted keys:

$$y^* = \boxed{\pi_*}(v^*\|r^*), \qquad s^* = \boxed{\sigma_*}(r^*),$$
$$P = \mathsf{TOF}[\boxed{\pi_*}]\|\mathsf{SCTOF}[\boxed{\delta^{\mathrm{rc}}_{x'}, \pi_*}]\|\mathsf{C'NOT}[\boxed{\delta^{\mathrm{rc}}_{x'}}, y^*]\|\mathsf{SCTOF}[\boxed{\delta^{\mathrm{rc}}_{x''}, \pi_*}],$$
$$Q = \mathsf{TOF}[\boxed{\sigma_*}]\|\mathsf{SCTOF}[\boxed{\delta^{\mathrm{rc}}_{r'}, \sigma_*}]\|\mathsf{C'NOT}[\boxed{\delta^{\mathrm{rc}}_{r'}}, s^*]\|\mathsf{SCTOF}[\boxed{\delta^{\mathrm{rc}}_{r''}, \sigma_*}],$$

$$R = \mathsf{TOF}[\boxed{\pi_*^{-1}}] \| \mathsf{SCTOF}[\delta^{\mathrm{rc}}_{y^*}, \boxed{\pi_*^{-1}}] \| \mathsf{SCTOF}[\boxed{\delta^{\mathrm{rc}}_{x'}\langle \pi_*^{-1}\rangle, \pi_*^{-1}}]$$
$$\| \mathsf{SCTOF}[\boxed{\delta^{\mathrm{rc}}_{x''}\langle \pi_*^{-1}\rangle, \pi_*^{-1}}] \| \mathsf{C'NOT}[\boxed{\delta^{\mathrm{rc}}_{x''}\langle \pi_*^{-1}\rangle}, v^* \| r^*].$$

By rc$i\mathcal{O}$ security, $\mathsf{G}_{2.5} \approx \mathsf{G}_3$.

- In G_4, we permute the keys by

$$\pi_* \xleftarrow{\$} \mathsf{PPRP.Permute}(\pi, (v^* \| r^* \;\; x' \;\; x''), \boxed{1}),$$
$$\sigma_* \xleftarrow{\$} \mathsf{PPRP.Permute}(\sigma, (r^* \;\; r' \;\; r''), \boxed{1}).$$

  We have $\mathsf{G}_3 \approx \mathsf{G}_4$ by PPRP security.

- In G_5, we sample

$$\mathcal{T}_0^{\mathrm{rc}} \xleftarrow{\$} \Delta^{\mathrm{rc}}(r^*), \qquad \mathcal{T}_{12}^{\mathrm{rc}} \xleftarrow{\$} \Delta^{\mathrm{rc}}(x''), \qquad \mathcal{T}_{22}^{\mathrm{rc}} \xleftarrow{\$} \Delta^{\mathrm{rc}}(r''),$$

  and update everything to use the triggers and the non-permuted keys:

$$y^* = \boxed{\pi(x')}, \qquad s^* = \boxed{\sigma(r')},$$
$$\boxed{P} = \mathsf{TOF}[\pi] \| \mathsf{SCTOF}[\boxed{\mathcal{T}_{12}^{\mathrm{rc}}}, \pi] \| \mathsf{SCTOF}[v^* \circledcirc \boxed{\mathcal{T}_0^{\mathrm{rc}}}, \pi] \| \mathsf{C'NOT}[v^* \circledcirc \boxed{\mathcal{T}_0^{\mathrm{rc}}}, y^*],$$
$$\boxed{Q} = \mathsf{TOF}[\sigma] \| \mathsf{SCTOF}[\boxed{\mathcal{T}_{22}^{\mathrm{rc}}}, \sigma] \| \mathsf{SCTOF}[\boxed{\mathcal{T}_0^{\mathrm{rc}}}, \sigma] \| \mathsf{C'NOT}[\boxed{\mathcal{T}_0^{\mathrm{rc}}}, s^*],$$
$$\boxed{R} = \mathsf{TOF}[\pi^{-1}] \| \mathsf{SCTOF}[\delta^{\mathrm{rc}}_{y^*}, \pi^{-1}] \| \mathsf{SCTOF}[\boxed{\mathcal{T}_{12}^{\mathrm{rc}}}\langle \pi^{-1}\rangle, \pi^{-1}].$$

  By rc$i\mathcal{O}$ security, $\mathsf{G}_4 \approx \mathsf{G}_5$.

- In $\mathsf{G}_{5.5}$, we remove the aborting condition (introduced in $\mathsf{G}_{2.5}$).
  We have $\mathsf{G}_5 \approx_{\mathrm{s}} \mathsf{G}_{5.5}$ at an error of no more than $6 \cdot 2^{-\lambda}$.

- In G_6, we deactivate the triggers by

$$\mathcal{T}_0^{\mathrm{rc}} \xleftarrow{\$} \boxed{\mathbb{1}_\lambda^{\mathrm{rc}}}, \qquad \mathcal{T}_{12}^{\mathrm{rc}} \xleftarrow{\$} \boxed{\mathbb{1}_{|v^*|+\lambda}^{\mathrm{rc}}}, \qquad \mathcal{T}_{22}^{\mathrm{rc}} \xleftarrow{\$} \boxed{\mathbb{1}_\lambda^{\mathrm{rc}}}.$$

  By trigger scheme security, $\mathsf{G}_5 \approx \mathsf{G}_6$.

- In G_7, we clean up the components:

$$y^* = \pi(x'), \qquad s^* = \sigma(r'),$$
$$P = \mathsf{TOF}[\pi] \| \cancel{\mathsf{SCTOF}[\mathcal{T}_{12}^{\mathrm{rc}}, \pi] \| \mathsf{SCTOF}[v^* \circledcirc \mathcal{T}_0^{\mathrm{rc}}, \pi] \| \mathsf{C'NOT}[v^* \circledcirc \mathcal{T}_0^{\mathrm{rc}}, y^*]},$$
$$Q = \mathsf{TOF}[\sigma] \| \cancel{\mathsf{SCTOF}[\mathcal{T}_{22}^{\mathrm{rc}}, \sigma] \| \mathsf{SCTOF}[\mathcal{T}_0^{\mathrm{rc}}, \sigma] \| \mathsf{C'NOT}[\mathcal{T}_0^{\mathrm{rc}}, s^*]},$$
$$R = \mathsf{TOF}[\pi^{-1}] \| \mathsf{SCTOF}[\delta^{\mathrm{rc}}_{y^*}, \pi^{-1}] \cancel{\| \mathsf{SCTOF}[\mathcal{T}_{12}^{\mathrm{rc}}\langle \pi^{-1}\rangle, \pi^{-1}]}.$$

  The removed operations always have their controls being false, so their action is the identity. By rc$i\mathcal{O}$ security, $\mathsf{G}_6 \approx \mathsf{G}_7$. Note that $\mathsf{G}_7 \equiv \mathsf{Exp}^1_{\text{cca-rr}}$ by renaming variables.

The proof concludes by hybrid argument. □

Remark 10 (adaptive security). Recall that to prove the adaptive security of our (normal-circuit) FE/PKE (Theorem 3), we rely on permutability by URS (see Lemma 1), which conditionally one-time-pads the plaintext and swaps the randomness. While URS can be implemented easily using reversible circuits, setting up the triggered program P turns out to be difficult.

The basic recipe of using IP-PRP is to pre-compose Γ^{-1} on the triggered branch so that the triggered branch reverts to normal after permuting the key by Γ (see Footnote 11), which boils down to

$$P(v\|r) = \begin{cases} \pi((u^* \oplus v)\|r^*), & \text{if } \mathcal{T}(r) = 1; \\ \pi(v\|r), & \text{otherwise;} \end{cases}$$

for adaptively secure PKE. It is unclear to us how to implement a Toffoli version of P *without ancillae* from π, $\mathcal{T}^{\mathrm{rc}}$, u^*, r^*. Notably, the activation point s^* of $\mathcal{T}^{\mathrm{rc}}$ is not known. It appears that one must place r^* on the wires, at least when the trigger is hit, so that π can be applied to compute the first branch. This seems hard as there is no sufficient space to stash r so that it can be eventually restored.

Conversely, the issue of space goes away once we have $|r| = \lambda$ ancillae initialized to zero. This can be done akin to the *hardened* Toffoli transformation (Remark 7). We choose to not present it due to its complexity.

Acknowledgments. The authors thank Claudio Chamon, Eduardo Mucciolo, and Andrei Ruckenstein for many enlightening discussions, including the idea of homomorphically evaluating a permutation f over encrypted data using the conjugation of f by the encryption permutation. We appreciate the constructive feedback from the anonymous reviewers of Crypto 2026. The authors were supported by NSF Grant GCR-2428487.

References

[AGVW13] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 500–518. Springer, Berlin, Heidelberg, August 2013.

[AS15] Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 191–209. IEEE Computer Society Press, October 2015.

[BBC+95] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A*, 52:3457–3467, November 1995.

[BBO07] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 535–552. Springer, Berlin, Heidelberg, August 2007.

[BCG+18] Nir Bitansky, Ran Canetti, Sanjam Garg, Justin Holmgren, Abhishek Jain, Huijia Lin, Rafael Pass, Sidharth Telang, and Vinod Vaikuntanathan. Indistinguishability obfuscation for RAM programs and succinct randomized encodings. *SIAM Journal on Computing*, 47(3):1123–1210, 2018.

[BFO08] Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 335–359. Springer, Berlin, Heidelberg, August 2008.

[BFOR08] Mihir Bellare, Marc Fischlin, Adam O'Neill, and Thomas Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 360–378. Springer, Berlin, Heidelberg, August 2008.

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *Journal of the ACM*, 59(2), May 2012.

[BKW17] Dan Boneh, Sam Kim, and David J. Wu. Constrained keys for invertible pseudorandom functions. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 237–263. Springer, Cham, November 2017.

[BL17] Johannes Blömer and Gennadij Liske. Subtleties in security definitions for predicate encryption with public index. In Johannes Blömer, Ilias S. Kotsireas, Temur Kutsia, and Dimitris E. Simos, editors, *Mathematical Aspects of Computer and Information Sciences*, pages 438–453. Springer, December 2017.

[BPW16] Nir Bitansky, Omer Paneth, and Daniel Wichs. Perfect structure on the edge of chaos - trapdoor permutations from indistinguishability obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 474–502. Springer, Berlin, Heidelberg, January 2016.

[Bro04] Alex Brodsky. Reversible circuit realizations of Boolean functions. In Jean-Jacques Levy, Ernst W. Mayr, and John C. Mitchell, editors, *Exploring New Frontiers of Theoretical Informatics*, volume 155 of *IFIP*, pages 67–80. Springer, 2004.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Berlin, Heidelberg, March 2011.

[CCMR24] Ran Canetti, Claudio Chamon, Eduardo R. Mucciolo, and Andrei E. Ruckenstein. Towards general-purpose program obfuscation via local mixing. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024, Part IV*, volume 15367 of *LNCS*, pages 37–70. Springer, Cham, December 2024.

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In Yevgeniy Dodis

and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 468–497. Springer, Berlin, Heidelberg, March 2015.

[CLZ26a] Ran Canetti, Ji Luo, and Yiding Zhang. How to encrypt with random reversible circuits: Functional, homomorphic and CCA-secure. Crypto 2026 (to appear), LNCS, Springer, 2026.

[CLZ26b] Ran Canetti, Ji Luo, and Yiding Zhang. Functional homomorphic encryption. Cryptology ePrint Archive (to appear), 2026.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

[GHP25] Lucas Gretta, William He, and Angelos Pelecanos. More efficient approximate k-wise independent permutations from random reversible circuits via log-sobolev inequalities. In Yossi Azar and Debmalya Panigrahi:, editors, *36th SODA*, pages 5582–5598. ACM-SIAM, January 2025.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th ACM STOC*, pages 365–377. ACM Press, May 1982.

[Gow96] W. T. Gowers. An almost m-wise independent random permutation of the cube. *Combinatorics, Probability and Computing*, 5(2):119–130, 1996.

[GWZ22] Jiaxin Guan, Daniel Wichs, and Mark Zhandry. Incompressible cryptography. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 700–730. Springer, Cham, May / June 2022.

[HJKS22] James Hulett, Ruta Jawale, Dakshita Khurana, and Akshayaram Srinivasan. SNARGs for P from sub-exponential DDH and QR. Cryptology ePrint Archive, Report 2022/353, 2022.

[IKO05] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Sufficient conditions for collision-resistant hashing. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 445–456. Springer, Berlin, Heidelberg, February 2005.

[IL25] Rahul Ilango and Alex Lombardi. Cryptography meets worst-case complexity: Optimal security and more from iO and worst-case assumptions. In *66th FOCS*, pages 585–614. IEEE Computer Society Press, December 2025.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger, editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Springer, 1972.

[KMO10] Eike Kiltz, Payman Mohassel, and Adam O'Neill. Adaptive trapdoor functions and chosen-ciphertext security. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 673–692. Springer, Berlin, Heidelberg, May / June 2010.

[Ms09] Steven Myers and abhi shelat. Bit encryption is complete. In *50th FOCS*, pages 607–616. IEEE Computer Society Press, October 2009.

[PD21a] Tapas Pal and Ratna Dutta. Chosen ciphertext secure functional encryption from constrained witness PRF. In Joonsang Baek and Sushmita Ruj, editors, *ACISP 21*, volume 13083 of *LNCS*, pages 24–45. Springer, Cham, December 2021.

[PD21b] Tapas Pal and Ratna Dutta. Chosen ciphertext secure functional encryption from constrained witness PRF. Cryptology ePrint Archive, Report 2021/512, version 20260415:061416.

[PR08] Manoj Prabhakaran and Mike Rosulek. Towards robust computation on encrypted data. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 216–233. Springer, Berlin, Heidelberg, December 2008.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 187–196. ACM Press, May 2008.

[RAD78] Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, pages 169–180, 1978.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, February 1978.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.

[SZ25] Omri Shmueli and Mark Zhandry. On one-shot signatures, quantum vs. classical binding, and obfuscating permutations. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part II*, volume 16001 of *LNCS*, pages 350–383. Springer, Cham, August 2025.

[VV85] Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. In *17th ACM STOC*, pages 458–463. ACM Press, May 1985.

[Wat15] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 678–697. Springer, Berlin, Heidelberg, August 2015.

A Proof of Lemma 2 (Random Triggers)

The existence of OWF immediately implies worst-case hardness of UniqueSAT and the proof of Lemma 17 (reversible-circuit random triggers) can be tweaked to construct a trigger scheme using any (non-reversible-circuit) $i\mathcal{O}$. However, that approach requires non-uniformly secure $i\mathcal{O}$ even if we only want uniformly secure triggers.

The proof in this section preserves (non-)uniform security. The trigger scheme is implicit in [BCG+18] but was abstracted as such. It uses an injective non-interactive commitment scheme, which can be based on any injective one-way function via Goldreich–Levin hardcore predicate. Injective OWF can be obtained from OWF and $i\mathcal{O}$, due to [BPW16]. (However, see Remark 11 on the subtleties.) Along the way, injectivity only has to hold with overwhelming probability, and this error is transferred to the advantage against the trigger scheme.

Definition 17 (injective non-interactive commitment). An *injective non-interactive commitment* consists of two efficient algorithms.

- $\mathsf{Gen}(1^\lambda)$ samples and outputs some public parameter pp.
- $\mathsf{Com}(1^\lambda, \mathsf{pp}, b; r)$ takes as input pp, a bit $b \in \{0,1\}$, and randomness $r \in \{0,1\}^\lambda$. It is deterministic and outputs some commitment com.

The scheme must be *(overwhelmingly) injective*, i.e.,

$$\Pr\left[\mathsf{pp} \xleftarrow{\$} \mathsf{Gen}(1^\lambda) \;:\; \begin{array}{c}\mathsf{Com}(1^\lambda, \mathsf{pp}, b_0; r_0) = \mathsf{Com}(1^\lambda, \mathsf{pp}, b_1; r_1) \text{ for some} \\ b_0, b_1 \in \{0,1\},\ r_0, r_1 \in \{0,1\}^\lambda \text{ with } (b_0, r_0) \neq (b_1, r_1)\end{array}\right] = \mathrm{negl}(\lambda).$$

The scheme is *secure* if

$$\left\{(1^\lambda, \mathsf{pp}, \mathsf{Com}(1^\lambda, \mathsf{pp}, 0; r))\right\}_{\lambda\in\mathbb{N}} \approx \left\{(1^\lambda, \mathsf{pp}, \mathsf{Com}(1^\lambda, \mathsf{pp}, 1; r))\right\}_{\lambda\in\mathbb{N}},$$

where $\mathsf{pp} \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$ and $r \xleftarrow{\$} \{0,1\}^\lambda$.

Lemma 21 ([BPW16,BCG+18]). *Based on one-way functions and $i\mathcal{O}$ (Definition 1), one can construct an injective non-interactive commitment scheme (Definition 17) with subexponential injectivity error. The construction preserves uniform security and "preserves" subexponential security (see Remark 11).*

We are ready to prove the trigger lemma.

Proof (Lemma 2, due to [BCG+18]). Let $i\mathcal{O}$ be an indistinguishability obfuscator and $(\mathsf{Gen}, \mathsf{Com})$ an injective non-interactive commitment scheme. The trigger scheme is as follows.

- $\not\Delta_N$ runs and outputs $i\mathcal{O}(\mathcal{O}_N)$. (Recall that $\mathcal{O}_N$ is the N-bit zero function.)
- $\Delta(r)$ runs and outputs $i\mathcal{O}(\delta_r)$. (Recall that δ_r is a point function.)

The paddings of $\mathcal{O}_N$, δ_r are determined by the circuit size of Com. The scheme is clearly correct.

For security, we focus on the case of $N \geq \lambda$ as $\mathsf{Exp}^\beta_{\mathrm{trigger}}$ outputs 0 when $N < \lambda$. Write P for the program being obfuscated into the output of the trigger scheme. Note that $r \xleftarrow{\$} \{0,1\}^N$ is sampled by $\mathsf{Exp}^\beta_{\mathrm{trigger}}$. We sample $\mathsf{pp} \xleftarrow{\$} \mathsf{Gen}()$ and let $r' \| r'' \leftarrow r$ with $|r''| = \lambda$. Consider the following hybrids, which are fully specified by P.

- H_0 is $\mathsf{Exp}^0_{\mathsf{trigger}}$, i.e., P is $P(x) = 0$.
- In H_1, we hook the commitment scheme into P as follows:

  $$\mathsf{com} = \mathsf{Com}(\mathsf{pp}, 0; r''),$$
  $$\boxed{P}(x) = \begin{cases} 1, & \text{if } x' \| \mathsf{Com}(\mathsf{pp}, 1; x'') = r' \| \mathsf{com}; \\ 0, & \text{otherwise;} \end{cases} \qquad \text{for } x' \| x'' \leftarrow x \text{ with } |x''| = \lambda.$$

  $\mathsf{Com}(\mathsf{pp}, \cdot; \cdot)$ is injective with overwhelming probability, in which case P still implements the all-zero function.

  We have $\mathsf{H}_0 \approx \mathsf{H}_1$ by $i\mathcal{O}$ security.
- In H_2, we switch com to be a commitment of 1, i.e.,

  $$\mathsf{com} = \mathsf{Com}(\mathsf{pp}, \boxed{1}; r''),$$
  $$P(x) = \begin{cases} 1, & \text{if } x' \| \mathsf{Com}(\mathsf{pp}, 1; x'') = r' \| \mathsf{com}; \\ 0, & \text{otherwise;} \end{cases} \qquad \text{for } x' \| x'' \leftarrow x \text{ with } |x''| = \lambda.$$

  By commitment security, $\mathsf{H}_1 \approx \mathsf{H}_2$.
- In H_3, we rewrite P into the point function:

  ~~$\mathsf{com} = \mathsf{Com}(\mathsf{pp}, 1; r'')$,~~ $\qquad \boxed{P}(x) = \begin{cases} 1, & \text{if } x = r; \\ 0, & \text{otherwise.} \end{cases}$

  $\mathsf{Com}(\mathsf{pp}, \cdot; \cdot)$ is injective with overwhelming probability, in which case, the condition $\mathsf{Com}(\mathsf{pp}, 1; x'') = \mathsf{Com}(\mathsf{pp}, 1; r'')$ is equivalent to $x'' = r''$ and the functionality of P is unchanged.

  We have $\mathsf{H}_2 \approx \mathsf{H}_3$ by $i\mathcal{O}$ security. Note that $\mathsf{H}_3 \equiv \mathsf{Exp}^1_{\mathsf{trigger}}$.

The advantage relation is $|\varepsilon_{\mathsf{trigger}}| \leq 2|\varepsilon_{i\mathcal{O}}| + 2\varepsilon_{\mathsf{inj}} + |\varepsilon_{\mathsf{Com}}|$, where $\varepsilon_{\mathsf{inj}}$ is the probability of Com being non-injective. When instantiated with Lemma 21, the trigger scheme is subexponentially secure, based on subexponentially secure OWF and $i\mathcal{O}$. □

For better concrete efficiency, a trigger with $N > \lambda$ can be reduced to one with λ-bit points, by putting the logic of comparison with the first $(N - \lambda)$ bits "in the clear", i.e., not obfuscated. For $\not\Delta$, the prefix comparison is with a random string.

Remark 11 (subexponential security in our work and [BPW16]). There are three popular notions of subexponential security:

- advantage is subexponential against subexponential-time adversaries, denoted "SA-ST";
- advantage is negligible against subexponential-time adversaries, "ST";
- subexponential, polynomial-time, "SA".

We have chosen SA-ST in our presentation.

The proof of Lemma 21 first obtains injective OWF from $i\mathcal{O}$ and plain OWF via [BPW16], then an injective commitment from injective OWF and Goldreich–Levin hardcore predicate [BCG+18]. (The second step retains the ingredients' security — SA-ST, ST, SA, polynomial.) The first step appears to *lose* SA (if unaccompanied by ST). In a bit more detail, the work of [BPW16] first constructs an injective *weak* OWF from $i\mathcal{O}$ and plain OWF, then amplifies it to an injective (strong) OWF. The nature of weak one-wayness disqualifies it from being SA. After amplification, the injective OWF is SA-ST if the ingredients are ST or (unnecessarily) SA-ST, and it is *(only) polynomially secure* if the ingredients are polynomially secure or *(even) SA*. As a result, we do not know how to base (SA version of) Lemma 2 on SA $i\mathcal{O}$ and plain OWF.

All our applications except Theorem 8 (unary HE) can be done with polynomial security. Below, we explain the case of unary HE.

- The advantage loss is determined by the number of hybrids (subexponential), so SA is always required.
- In the setting of uniform security, the reductions can take up to subexponential extra time ("extra" on top of the underlying adversary), so SA-ST is needed.
- For non-uniform security, suppose we are given SA triggers (or SA injective OWF), then all the reductions only take polynomial extra time and zero or polynomially long extra advice, so we only need SA $i\mathcal{O}$. However, if we start with plain OWF, then both $i\mathcal{O}$ and plain OWF need to be SA-ST.