New Random Oracle Instantiations from Extremely Lossy Functions

Chris Brzuska[1], Geoffroy Couteau[2], Christoph Egger[2], Pihla Karanko[1], and Pierre Meyer[2,3]

[1] Aalto University, Finland. {chris.brzuska,pihla.karanko}@aalto.fi
[2] Université Paris Cité, CNRS, IRIF, France. {couteau,christoph.egger,pierre.meyer}@irif.fr
[3] Reichman University, Israel.

Abstract. We instantiate two random oracle (RO) transformations using Zhandry's extremely lossy function (ELF) technique (Crypto'16). Firstly, using ELFs and indistinguishabililty obfuscation (iO), we instantiate a modified version of the Fujisaki-Okamoto (FO) transform which upgrades a public-key encryption scheme (PKE) from indistinguishability under chosen plaintext attacks (IND-CPA) to indistinguishability under chosen ciphertext attacks (IND-CCA). We side-step a prior uninstantiability result for FO by Brzuska, Farshim, and Mittelbach (TCC'15) by (1) hiding the randomness from the (potentially ill-designed) IND-CPA encryption scheme and (2) embedding an additional secret related to the hash-function into the secret-key of the IND-CCA-secure PKE, an idea brought forward by Murphy, O'Neill, Zaheri (Asiacrypt 2022) who also instantiate a modified FO variant also under ELFs and iO for the class of *lossy PKE*. Our transformation applies to all PKE which can be inverted given their randomness. Secondly, we instantiate the hash-then-evaluate paradigm for pseudorandom functions (PRFs), $\mathsf{PRF}_{\mathsf{new}}(k, x) := \mathsf{wPRF}(k, \mathsf{RO}(x))$. Our construction replaces RO by $\mathsf{PRF}_{\mathsf{old}}(k_{\mathsf{pub}}, \mathsf{elf}(x))$ with a key k_{pub}, that, unusually, is known to the distinguishing adversary against $\mathsf{PRF}_{\mathsf{new}}$. We start by observing that several existing weak PRF candidates are plausibly also secure under such distributions of *pseudorandom* inputs, generated by $\mathsf{PRF}_{\mathsf{old}}$. Firstly, analogous cryptanalysis applies and/or an attack with such pseudorandom inputs would imply surprising results such as key agreement from the high-noise version of the Learning Parity with Noise (LPN) assumption. Our simple transformation applies to the entire family of PRF-style functions. Specifically, we obtain results for oblivious PRFs, which are a core building block for password-based authenticated key exchange (PAKE) and private set intersection (PSI) protocols, and we also obtain results for pseudorandom correlation functions (PCF), which are a key tool for silent oblivious transfer (OT) extension.

Keywords: random oracle model, extremely lossy functions, Fujisaki-Okamoto transform, pseudorandom functions, pseudorandom correlation functions

1 Introduction

The random oracle model (ROM) [BR93] is an idealised security model where all parties, honest or otherwise, are given the same uniformly chosen random function as an oracle. Random oracles (ROs) model ideal hash functions and have found a plethora of applications in cryptography, including the Fiat-Shamir [FS87] transformation from 3-round interactive to non-interactive zero-knowledge proofs (NIZK), key-dependent message (KDM) security [BRS03], adaptively secure garbled circuit [BHR12] and many more. In this work, we are particularly interested in (1) RO-based transforms for turning IND-CPA-secure PKE into IND-CCA-secure PKE as in the Fujisaki-Okamoto transform (FO) [FO13], as well as (2) RO-based pre-processing of inputs as used, *e.g.* for password-based authenticated key exchange (PAKE) [CHL22a] and private set intersection (PSI) constructions [HL08]. Concretely, both PAKE and PSI first pre-process their inputs—a password for PAKE and database entries for PSI—by applying a RO and then use secure multi-party computation to evaluate a weak PRF on the RO result, a so-called *oblivious PRF* (OPRF) evaluation. This *hash-then-evaluate* paradigm, thus, pushes some of the complexity of the PRF $\mathsf{PRF}(k, x) := \mathsf{wPRF}(k, \mathsf{RO}(x))$ into a purely offline phase, outside of the 2PC.

RO-based proofs for FO and the hash-then-evaluate paradigm make heavy use of the fact that the reduction emulates the random oracle to the adversary and can therefore *observe* all queries to the RO as well as *program* the RO. In effect, the reduction chooses the mapping of the RO adaptively during the security experiment.

1.1 RO (Un)instantiability

The FO transform is a core building block for the post-quantum secure key encapsulation schemes standardised by NIST [NIS16] and the hash-then-evaluate paradigm is at the heart of the currently most efficient password-based authenticated key (PAKE) protocols and PSI schemes (cf. [CHL22a, DGH$^+$21]).

Despite the practical relevance of FO and the hash-then-evaluate paradigm, we do not know how to instantiate the RO in the two transforms. Even worse, a strong uninstantiability result by Brzuska, Farshim and Mittelbach (BFM [BFM15]) shows that the random oracle in FO is actually *uninstantiable*. Namely, BFM build an IND-CPA-secure PKE based on indistinguishability obfuscation (iO) such that its FO transform $\mathsf{FO}^{\mathsf{RO}_1,\mathsf{RO}_2}[\mathsf{PKE},\mathsf{SE}]$ is IND-CCA *in the ROM*, but becomes insecure as soon as RO_1 and RO_2 are replaced by any concrete hash function (distribution).

Positive results. However, similarly to FO, the Fiat-Shamir (FS) transform also suffers from general uninstantiability results [GK03,BDG$^+$13] and yet, via correlation-intractable hash functions [CGH98], a highly successful research line instantiates FS for *specific* proof systems (which is not ruled out by the uninstantiability results), leading to new efficient NIZKs [CKU20], new constructions of SNARGs and batch arguments [JKKZ21], insights on PPAD hardness [BCH$^+$22], and more. Another line of research circumvents impossibility results via the *2-stage assumption* of Universal Computational Extractors [BHK13], resulting in KDM-secure encryption, and adaptive garbled circuits, among others.

Last, but not least, Zhandry [Zha16] introduces the brillant *non-black-box* framework of *extremely lossy functions (ELFs)* to build secure point function obfuscation with auxiliary input, polynomially-many hardcore bits for any one-way function and output intractable hash function—all inherently hard in the standard model—and later also deterministic encryption [Zha19].

ELF. An ELF can be sampled either to be injective or lossy with a $\mathsf{poly}(\lambda)$-size image, and yet, the injective mode and the lossy modes are indistinguishable—if the adversary $\mathcal{A}$ is bounded by a *fixed* polynomial $\mathsf{poly}'(\lambda)$ as long as $\mathsf{poly}'(\lambda) \ll \mathsf{poly}(\lambda)$ and $\frac{1}{\mathsf{poly}(\lambda)} \ll \mathsf{Adv}(\mathcal{A})$, where $\mathsf{Adv}(\mathcal{A})$ denotes the adversary's advantage. Since the notion is central to this paper (and so the introduction remains self-contained), we now formally define *extremely lossy functions.*

Definition 1 (Extremely Lossy Function (ELF), adapted from [Zha16]). *An* Extremely Lossy Function *(*ELF*) is a PPT algorithm* $\mathsf{ELF.Gen}$ *which, on inputs* 1^λ *and* $r \in [2^\lambda]$*, outputs a polynomial-time computable*[4] *function* $\mathsf{elf}\colon \{0,1\}^\lambda \to \{0,1\}^*$ *such that the following hold:*

Injectivity. $\Pr_{\mathsf{elf}\leftarrow_\$\mathsf{ELF.Gen}(1^\lambda,2^\lambda)}\left[|\mathsf{elf}(\{0,1\}^\lambda)| = 2^\lambda\right] = 1 - \mathsf{negl}(\lambda)$.

Lossiness. $\forall c_{\mathsf{elf}} \in \mathbb{N}$*:* $\Pr_{\mathsf{elf}\leftarrow_\$\mathsf{ELF.Gen}(1^\lambda,\lambda^{c_{\mathsf{elf}}})}\left[|\mathsf{elf}(\{0,1\}^\lambda)| \leq \lambda^{c_{\mathsf{elf}}}\right] = 1 - \mathsf{negl}(\lambda)$.

Indistinguishability. $\forall a,t \in \mathbb{N}, \exists c \in \mathbb{N}$ *such that for all* $\mathcal{A}$ *running in time* $\leq \lambda^t$

$$\left|\Pr_{\mathsf{elf}\leftarrow_\$\mathsf{ELF.Gen}(1^\lambda,2^\lambda)}\left[1 = \mathcal{A}(1^\lambda,\mathsf{elf})\right] - \Pr_{\mathsf{elf}\leftarrow_\$\mathsf{ELF.Gen}(1^\lambda,\lambda^c)}\left[1 = \mathcal{A}(1^\lambda,\mathsf{elf})\right]\right| < \lambda^{-a} \ .$$

Enumerable image. $\forall c_{\mathsf{elf}} \in \mathbb{N}$*,* $\exists$ *PPT*$(\lambda^{c_{\mathsf{elf}}})$*,* $\mathcal{C}$: $\Pr_{\mathsf{elf}\leftarrow_\$\mathsf{ELF.Gen}(1^\lambda,\lambda^{c_{\mathsf{elf}}})}\left[\mathsf{elf}(\{0,1\}^\lambda) \subseteq \mathcal{C}(1^\lambda,\mathsf{elf})\right] \geq 1 - \mathsf{negl}(\lambda)$

The *non-black-box property* (via dependency on the adversary's runtime) as well as the polynomial-time enumerability of the image space are two powerful tools for instantiating random oracles.

1.2 Contribution

New RO instantiations in the standard model. Our first main contribution uses ELFs and indistinguishability obfuscation to instantiate a slightly modified FO transform, side-stepping the BFM impossibility result. We present a high-level technical overview of this construction in Section 2, and provide the details in Section 6.

[4] We here refer to a polynomial in λ, and this polynomial is global for all elf which are returned by $\mathsf{ELF.Gen}(1^\lambda,*)$

Our second main contribution is to instantiate the hash-and-evaluate paradigm for a wide range of "PRF-like" objects. We start with instantiating this approach for PRFs, which has direct implications for low-complexity OPRFs. We then move on to boosting *pseudo-correlation functions* (PCFs)[5] from weak to strong via the hash-then-evaluate paradigm. PCFs were introduced by Boyle, Couteau, Gilboa, Ishai, Kohl, and Scholl [BCG$^+$20] and allow two parties[6] who each hold one of two short correlated seeds k_0 and k_1 to generate any polynomial number of *correlated* randomess by evaluating $\mathsf{pcf}(k_0, x_i)$ and $\mathsf{pcf}(k_1, x_i)$ *locally* on the same public inputs x_i such that the outputs are indistinguishable from the target correlation. Thus, PCFs allow to implement Beaver's OT extension idea [Bea96] in a *silent* manner, that is, without additional communication.

Revisiting the analysis of wPRFs when inputs are pseudorandom. Our instantiation of hash-then-evaluate replaces the random oracle by the hash-function $\mathsf{H}(.) := \mathsf{PRF}(k_\mathsf{H}, \mathsf{elf}(.))$ with a *public* PRF key k_H. That is, we replace the RO by a *public* pre-processing phase of the input so that $\mathsf{wPRF}(k, \mathsf{H}(.))$ is a strong PRF, but the pre-processing does not depend on k and thus, in secure multi-party computation applications, only the second wPRF part needs to be securely evaluated.

One caveat is that the outputs of $\mathsf{H}(.)$ are not random and wPRF expects random inputs. However, when k_H is not given, they are at least *pseudorandom.* Our third main result is the observation that the cryptanalysis of several weak PRFs [BIP$^+$18,BCG$^+$20,BCG$^+$22] actually also applies when the inputs are pseudorandom, namely, for as long as they satisfy suitable statistical properties—or at least, that violating the security of these wPRFs for *pseudorandom* inputs would have interesting consequences. In slightly more details, we consider the weak PRF of [BIP$^+$18], which is at the heart of the most efficient (to date) oblivious PRF protocol [DGH$^+$21], and the weak PRFs of [BCG$^+$20, BCG$^+$22], which are at the heart of the most efficient PCFs known to date. For each of these candidates, we analyze the most natural families of attacks, and obtain the following results:

- The wPRF candidate of [BIP$^+$18] is secure against all *statistical query algorithms* (one of the most common types of attacks against low-complexity PRF candidates, used *e.g.* to show that there are no subexponentially secure wPRFs in AC^0 [LMN89]) even when the wPRF inputs are pseudorandom (instead of truly random). This extends the analysis of [BIP$^+$18], which focused on truly random inputs.
- Either the wPRF candidates of [BCG$^+$20,BCG$^+$22] are secure against attacks from the *linear test framework* (the main framework used to study the security of "LPN-style" wPRFs, which both these candidates are) even when the wPRF inputs are pseudorandom, or there exists an efficiently samplable family of linear codes which have large (linear) *pseudodistance* (*i.e.* are computationally indistinguishable from codes with a linear distance) but low (sublinear) minimum distance. The existence of such codes is an open question which would have interesting consequences for low-complexity cryptographic hash functions [AHI$^+$17].

We expand on our analysis in Section 5. Given the background that several wPRFs are plausibly secure against pseudorandom inputs, we introduce the concept of pseudorandom-input PRFs (PI-PRFs) and pseudorandom input PCFs (PI-PCFs), and show that the above transform provably turns PI-PRFs into strong PRFs. We here rely on the the property that in extremely lossy mode of the elf, we can efficiently enumerate over all possible queries $q = \mathsf{H}(x)$ to the PI-PRF. Our core observation is that the set of values $\mathsf{Im}(\mathsf{H}) = \mathsf{PI\text{-}PRF}(k_\mathsf{H}, \mathsf{Im}(\mathsf{elf}))$ is pseudorandom, since $\mathsf{Im}(\mathsf{elf})$ does not depend on the PRF key.

Organization. We start with a technical overview of our FO instantiation in Section 2. Then, we turn to PRFs and OPRFs as our technical warm-up for the hash-then-evaluate paradigm instantiation in Section 3, which also provides additional context, motivation and comparison to related work, notably the closely related concept of public-coin PRFs by Pietrzak and Sjödin [PS08], which inspires our work. We then apply out hash-then-evaluate technique to boost PCFs from weak to strong in Section 4. We present our cryptanalysis and reflection on the plausibility of existing wPRF / wPCFs being *pseudorandom-input* PRFs / PCFs in Section 5 and the technical details of the FO instantiation in Section 6.

[5] Sometimes described as a *Correlated Pseudorandom Functions.*

[6] For simplicity of exposition, we only consider two-party PCFs, but the hash-then-evaluate is also compatible with the multiparty setting.

2 Technical Overview of FO Instantiation

Fujisaki-Okamoto transformation. FO turns any IND-CPA-secure PKE and any 1-time-secure authenticated encryption scheme SE into an IND-CCA-secure encryption scheme $\mathsf{FO}^{\mathsf{RO}_1,\mathsf{RO}_2}[\mathsf{PKE},\mathsf{SE}]$ which uses two random oracles RO_1 and RO_2 as follows:

$$\mathsf{FO}^{\mathsf{RO}_1,\mathsf{RO}_2}[\mathsf{PKE},\mathsf{SE}].\mathsf{Enc}(\mathsf{pk},m;\sigma) := (\mathsf{PKE}.\mathsf{Enc}(\mathsf{pk},\sigma;\mathsf{RO}_1(\sigma||m)),\mathsf{SE}.\mathsf{Enc}(\mathsf{RO}_2(\sigma),m)) \tag{1}$$

The decryption recovers σ from the PKE ciphertext, computes $\mathsf{RO}_2(\sigma)$ to obtain m from the SE ciphertext and then re-encrypts to check whether the same ciphertext is obtained. The FO reduction to IND-CPA of PKE observes the adversary's queries to RO_1 to determine all possible message and randomness candidates $\sigma||m$ and can thereby simulate answers to the decryption oracle.

BFM uninstantiability result. BFM construct their counterexample $\mathsf{PKE}_{\mathsf{BFM}}$ from an IND-CPA-secure scheme $\mathsf{PKE}_{\mathsf{cpa}}$ by appending an obfuscated circuit $C[\sigma,r_{\mathsf{cpa}}]$, which has σ and the randomness r_{cpa} of $\mathsf{PKE}_{\mathsf{cpa}}.\mathsf{Enc}$ hardcoded and works roughly as follows:

Inputs: candidate hash-functions $\mathsf{H}_1,\mathsf{H}_2$
symmetric ciphertext c_{sym}

Computation: $m \leftarrow \mathsf{SE}.\mathsf{Dec}(\mathsf{H}_2(\sigma),c_{\mathsf{sym}})$
if $\mathsf{H}_1(\sigma)=r_{\mathsf{cpa}}$ **then return** m **else return** $\perp$

Our construction. The BFM impossibility result needs that (i) σ is input to the PKE, and (ii) σ, H_1 and H_2 suffice to obtain randomness r_{cpa}. In order to circumvent the BFM impossibility result, (i) we encrypt $\mathsf{H}_1(\sigma)$ instead of σ and (ii) encrypt m with $\mathsf{H}_2(\sigma)$. Additionally, H_1 takes σ instead of $\sigma||m$ as input since σ is uniformly random and thus a suitable input to a pseudorandom generator (PRG). This change is w.l.o.g., since H_1 could also just ignore m. We obtain the following modified transform:

$$\mathsf{FO}_{\mathsf{mod}}^{\mathsf{H}_1,\mathsf{H}_2}[\mathsf{PKE},\mathsf{SE}].\mathsf{Enc}(\underbrace{\mathsf{pk}}_{:=(\mathsf{pk}_{\mathsf{cpa}},\mathsf{H}_1,\mathsf{H}_2)},m;\sigma) := (\mathsf{PKE}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cpa}},\mathsf{H}_2(\sigma);\mathsf{H}_1(\sigma)),\mathsf{SE}.\mathsf{Enc}(\mathsf{H}_2(\sigma),m)) \tag{2}$$

Decryption can still be correctly performed by recovering $k_{\mathsf{sym}} := \mathsf{H}_2(\sigma)$ and decrypting the symmetric ciphertext. However, *re-encryption* requires recovering σ, and the PKE ciphertext only contains $\mathsf{H}_2(\sigma)$ instead of σ itself. Therefore—and that is the bigger change—our construction samples H_1 and H_2 *together* with $(\mathsf{sk},\mathsf{pk})$ and then (i) includes H_1 and H_2 into the public-key (cf. (2)), and (ii) embeds a secret into the secret-key such that given $\mathsf{H}_2(\sigma)$, the decryptor can recover $\mathsf{H}_1(\sigma)$ as well. Concretely, the hash-function H_1 consists of an indistinguishability obfuscation (iO) (cf. Definition 41) of

$$\mathsf{PPRF}(k_{\mathsf{PPRF}},\mathsf{elf}(\mathsf{H}_2(\cdot))),$$

i.e., H_1 first applies H_2 to its input, then an elf and then a puncturable pseudorandom function (PPRF) (cf. Definition 40). Naturally, when the secret key includes k_{PPRF} and elf, the decryption can compute $\mathsf{H}_1(\sigma)$ given $\mathsf{H}_2(\sigma)$.

Our security reduction to IND-CPA-security of our modified FO-transform then uses that the elf in lossy mode has a polynomial-size, *enumerable* image space,. Therefore, we can enumerate over all possible randomness values $\mathsf{H}_1(\sigma)$ that might have been used in encryption. If given the randomness, encryption of PKE is invertible (more on this point shortly), then we can use inversion of encryption to simulate the decryption oracle.

One caveat in this argument is that neither the decryption algorithm nor its simulation can check whether $k_{\mathsf{sym}}=\mathsf{H}_2(\sigma)$ is actually *valid*, i.e., whether a given value k_{sym} is indeed in the image of H_2 or not. This is a problem since we'd like to use H_2 as a PRG together with puncturing in Sahai-Waters-style [SW14], and since we need to puncture the key both in the (obfuscated) code of H_2 and in the decryption oracle, it is a problem, if the decryption algorithm evaluates the PPRF also outside of the image of $\mathsf{H}_2=\mathsf{PRG}$. Thus, for H_2, we do not use an ordinary PRG, but rather a PRG where we can check image membership using a secret-key. Such a PRG, in fact, is the same as a \$-IND-CCA-secure encryption scheme, namely, we define

$$\mathsf{H}_2(\sigma) := \mathsf{PKE}_{\mathsf{cca}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cca}},w;\sigma)$$

for a *fixed* message w and a fixed public-key $\mathsf{pk}_{\mathsf{cca}}$, treating the input to H_2 as the randomness of $\mathsf{PKE}_{\mathsf{cca}}.\mathsf{Enc}$. Although it is a bit counter-intuitive to use an IND-CCA-secure encryption scheme to prove security of a transform for IND-CCA-security, it is meaningful for a feasibility result which, otherwise, already relies on the very strong assumptions of ELFs and iO. Importantly, the construction uses IND-CCA-secure encryption *inside* the IND-CPA-secure PKE, so that the non-malleability of the combined scheme does not result from the underlying IND-CCA-secure scheme. For example, if $\mathsf{PKE}_{\mathsf{cpa}}$ encapsulates a key k_{cpa} and encrypts its payload x as $k_{\mathsf{cpa}} \oplus x$, then the adversary can still make arbitrary additive changes to the message. Similarly, if the CPA scheme adds some redundant bits to the ciphertext, the adversary can still tamper with those.

Invertibility given randomness. Our transform works for all IND-CPA-secure PKE that are invertible when given the randomness (Definition 37). This property is naturally satisfied by many encryption schemes. E.g., given the exponent r for g^r in El Gamal, one can compute pk^r and thus recover the blinding value. Similarly, given the noise value e, Learning Parity with Noise (LPN) and Learning with Errors (LWE) both become easy. In fact, one can also simply transform every PKE into an invertible one by (1) doubling the length of the randomness σ into σ and σ' and then concatenating $m \oplus \sigma'$ to the ciphertext. Since this is, information-theoretically, only a uniformly random string, all other security properties of the PKE are preserved.

Comparison with MOZ. Murphy, O'Neill, Zaheri (MOZ) [MOZ22] gave an instantiation of an FO-like transform which turns a lossy PKE into an IND-CCA one. Similar to our instantiation, MOZ also include hash-function related secrets into the key of the new PKE. Interestingly, the MOZ construction is *dual* to ours in that the complexity of their hash-functions is swapped compared to ours. MOZ provide a complex—and very similar—instantiation of H_1 and instantiate H_2 simply by a pairwise independent hash-function. This is possible in the MOZ work because the PKE is *lossy* and thus, after switching to lossy mode, there is sufficient real randomness for both, the PKE and the key k_{sym} which the PKE encrypts. A minor difference between MOZ and our work is that MOZ needs exponential security on additional primitives, including the iO, if the message is allowed to depend on the public key. In turn, we only rely on polynomial security—except for Zhandry's ELF construction which inherently relies on exponential DDH.

Practicality. Finally, we remark that both MOZ and our work are intended to chart the territory to see for which classes of PKE FO-like transforms can actually be based on established cryptographic assumptions. MOZ show that the relevant class of lossy encryption schemes is included, and our result for randomness-invertible schemes directly comprises most practically and theoretically considered schemes in the literature. However, both MOZ and our result still require significant follow-up work in order to derive practically efficient instantiations, our complex hash functions are not efficient, but they show a promising way around the (very general!) BFM impossibility result.

2.1 Notation

In the following we will also make heavy use of the following notation. We write $\mathrm{PPT}(\lambda)$ for probabilistic polynomial time where the time is polynomial in λ. Often, we just write PPT and the security parameter λ is implicit. $x \leftarrow\!\!\$\, S$ denotes sampling a variable x uniformly from the set S and analogously $x \leftarrow\!\!\$\, \mathcal{A}$ denotes that the variable x is chosen by a PPT algorithm $\mathcal{A}$ with implicit randomness. For assigning a variable x according to a (not necessarily efficiently sampleable) distribution $\mathcal{D}$, we write $x \hookleftarrow \mathcal{D}$. When the randomness is passed explicitly to an algorithm we separate it by a semicolon (;) as in $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m\,;r)$. We use $y \leftarrow f(x)$ to emphasize that the assignment is deterministic. Finally, when an algorithm $\mathcal{A}$ has access to an oracle ORACLE we annotate it as $\mathcal{A}^{\mathsf{ORACLE}}$.

3 Technical Warm-Up: Instantiating Hash-then-Evaluate PRFs

The *Hash-then-Evaluate* paradigm relies on the fact that, if wPRF is a weak PRF and RO is a *programmable* random oracle, then $\mathsf{wPRF} \circ \mathsf{RO}$ is a strong PRF. This transformation truly shines in distributed settings (*e.g.* distributed PRFs, oblivious PRFs, and correlated PRFs—better known as pseudorandom correlation functions—), where the hash function is applied locally in an *input pre-processing phase*, thereby limiting the use of expensive compilers (such as *secure multiparty computation*) to securely evaluate only *weak* PRFs, which admit significantly lower-complexity candidates than strong PRFs.

The goal of this section is to provide a gentle introduction to our instantiation techniques in the simple setting of PRFs, without the additional difficulties tied to distributed considerations. Two issues emerge when we try to instantiate RO in this weak-to-strong PRF transform using an *extremely lossy function.* The first is conceptual, but the second is merely a technical inconvenience.

First issue: Programmability. The core reason programmability is required in the transformation $\mathsf{PRF} := \mathsf{wPRF} \circ \mathsf{RO}$ is the fundamental mismatch between the security games of weak and strong PRFs. In the former, the evaluation points are sampled in the experiment and delivered as output to the distinguisher, while in the latter the distinguisher provides its own queries. If we try and prove *strong PRF security* of $\mathsf{wPRF} \circ \mathsf{RO}$ and wish to invoke the *weak PRF security* of wPRF, it seems the reduction needs to *first* sample the value y and *then*, upon receiving an adversarial query x, ensure that $y = \mathsf{RO}(x)$.

The first issue is that our instantiation techniques seem ill-suited to achieve such a strong notion of programmability. Indeed, our approach is to first apply an *extremely lossy function* so as to force a non-adaptive adversaries' queries to be restricted to a polynomial-size set[7]—which is efficiently enumerable, but not necessarily programmable—, and then apply a non-adaptive PRF in order to (pseudo)randomise the image of the ELF.

We resolve this issue by weakening the scope of the transform, not starting from a weak PRF, but from a slightly stronger notion, whose security game has the adversary sample the queries (but these queries should be forced to remain "close enough to random" in order to not stray to far from the goal of boosting weak PRFs to strong ones). Specifically, we introduce the notion of *pseudorandom-input PRF* (PI-PRF), whose security is expected to hold if it is evaluated on points which are not truly random but rather *sampled pseudorandomly according to a public seed.*

Second "issue": The need for a CRS. We provide an instantiation of the random oracle which turns a PI-PRF into a strong PRF. Unfortunately, since we do not provide a fixed hash function H but rather a family of hash functions $(\mathsf{H}_r)_r$, the resulting construction is not a strong PRF, but rather a *strong PRF in the Common Random String model.* It may seem that we could get rid of the CRS by making r part of the key: $\mathsf{sPRF}(k_{\mathsf{sPRF}} = (k_{\mathsf{wPRF}}, r), x) := \mathsf{wPRF}(k_{\mathsf{wPRF}}, \mathsf{H}_r(x))$. This solution is unsatisfactory as it obfuscates the fact security of the resulting strong PRF holds if r is public, and additionally does not accurately model the fact that applying H_r is merely client-side input preprocessing. Using a CRS is not a problem *per se* (especially when the alternative is the random oracle model, which essentially already assumes the existence of globally accessible setup), but it makes formal theorem statements a bit awkward. In this section, which only deals with boosting the security of PRFs in a centralised setting, we will mostly sweep the CRS formalism under the rug so as to not distract away from the core ideas behind the instantiation. When instantiating hash-then-evaluate OPRFs and sPCFs however, which are our main results, we will fully address this technicality.

Very concretely our instantiation consists in a non-adaptive PRF, with its key hardcoded, composed with an injective-mode ELF. As a side benefit, this transformation is modular in nature: at a high level, only applying a non-adaptive PRF (with a hard-coded key) boosts security from pseudorandom-input to non-adaptive, and applying the ELF then boosts security from non-adaptive to adaptive. Because the first step is more lightweight (both in terms of efficiency and assumptions) than the second, this is interesting in settings where adaptive security is not required. Unfortunately, because of the aforementioned CRS technicality, things are not as simple as a transformation from PI-PRF to non-adaptive PRF (in the CRS model) to strong PRF (in the CRS model); instead, the intermediary object is not the standard notion of non-adaptive PRF but rather a "fully non-adaptive PRF (in the CRS model)", whose security only holds if the evaluation points are chosen non-adaptively and *before seeing the CRS.* This intermediary notion has the downside of being less standard, but it still captures the useful notion of security *provided evaluation points are selected according to a pre-agreed upon order.* In particular, this is already how non-adaptive PCFs are used, so this restriction is of little importance when it comes to MPC applications of our techniques.

[7] More precisely, the construction always uses the ELF in injective mode, so this "restriction" is purely hypothetical, and done only in the security proof.

3.1 Pseudorandom-Input PRF (PI-PRF)

In this section we introduce the notion of a *pseudorandom-input PRF* (PI-PRF), which can be seen as a strengthening of a wPRF. Informally, a PI-PRF is a wPRF which remains secure if queried on pseudorandom (instead of truly random) evaluation points, where the seed used as a source of pseudorandomness is public. In order to define PI-PRF, we first need to define a sampler that produces the "random looking" inputs. We call such samplers *admissible*, and provide a formal definition in Definition 2.

Definition 2 (Admissible Sampler). *We say that a polynomial time sampler* $\mathsf{Sam}_{\lambda,N} : \{0,1\}^{\mathsf{len}} \to \{0,1\}^{N \times \lambda}$*, where* len *is polynomial in* λ*, is* admissible *if for all PPT* $\mathcal{A}$

$$|\Pr_{r \leftarrow \$ \{0,1\}^{\mathsf{len}}}[1 = \mathcal{A}(\mathsf{Sam}_{\lambda,N}(r))] - \Pr_{\forall i : x_i \leftarrow \$ \{0,1\}^{\lambda}}[1 = \mathcal{A}(x_1, ..., x_N)]| = \mathit{negl}(\lambda).$$

Sometimes we might not write the randomness r *explicitly, but instead consider* $\mathsf{Sam}_{\lambda,N}$ *as PPT adversary that samples* r *uniformly itself and does not take any input. We write* $\mathit{inLen}(\mathsf{Sam}_{\lambda,N}) := |r|$ *for the length of the randomness.*

We can now define a pseudorandom-input PRF. Using the terminology of Pietrzak and Sjödin [PS08], our notion of PI-PRF could alternatively be defined as a *public coin wPRF with admissible samplers (as defined in Definition 2)*[8]. Note that in the definition of admissible sampler, the adversary $\mathcal{A}$ does not see the sampler's randomness r (only Sam output), but, in the definition of PI-PRF, the adversary $\mathcal{A}$ is given r.

Definition 3 (Pseudorandom Functions (PRF)). *Let* $\lambda \in \mathbb{N}$ *denote a security parameter. A pseudorandom function is syntactically defined as a collection of functions* $f_\lambda \colon \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$*, where the output* $f_\lambda(K, x)$ *can be computed from* (K, x) *in polynomial time. When unambiguous, we we will drop the* λ *index. We say that* f *is a:*

- **strong PRF [GGM84]** *if for all* p.p.t. $\mathcal{A}^{\mathsf{EVAL}}$ *who never queries the same input* x *twice to the oracle* EVAL*:*

$$\left|\Pr\left[1 = \mathsf{Exp}^{\mathsf{sPRF}}_{\mathcal{A},f,0}\right] - \Pr\left[1 = \mathsf{Exp}^{\mathsf{sPRF}}_{\mathcal{A},1}\right]\right| = \mathsf{negl}(\lambda),$$

  where $\mathsf{Exp}^{\mathsf{sPRF}}_{\mathcal{A},f,0}$ *and* $\mathsf{Exp}^{\mathsf{sPRF}}_{\mathcal{A},1}$ *are defined in Figure 1a.*

- **non-adaptive PRF [NR95]**[9] *for all* p.p.t. $\mathcal{A}$ *which output a vector* $\vec{x}$ *of unique input values* x

$$\left|\Pr\left[1 = \mathsf{Exp}^{\mathsf{naPRF}}_{\mathcal{A},f,0}\right] - \Pr\left[1 = \mathsf{Exp}^{\mathsf{naPRF}}_{\mathcal{A},1}\right]\right| = \mathsf{negl}(\lambda),$$

  where $\mathsf{Exp}^{\mathsf{naPRF}}_{\mathcal{A},f,0}$ *and* $\mathsf{Exp}^{\mathsf{naPRF}}_{\mathcal{A},1}$ *are defined in Figure 1b.*

- **weak PRF [NR95]** *for all polynomials* N*, for all* p.p.t. $\mathcal{A}$

$$\left|\Pr\left[1 = \mathsf{Exp}^{\mathsf{wPRF}}_{N,\mathcal{A},f,0}\right] - \Pr\left[1 = \mathsf{Exp}^{\mathsf{wPRF}}_{N,\mathcal{A},1}\right]\right| = \mathsf{negl}(\lambda),$$

  where $\mathsf{Exp}^{\mathsf{wPRF}}_{p,\mathcal{A},f,0}$ *and* $\mathsf{Exp}^{\mathsf{wPRF}}_{p,\mathcal{A},1}$ *are defined in Figure 1c.*

- **pseudorandom-input PRF [this paper]** *if for all polynomials* N*, for all admissible samplers* $\mathsf{Sam}_{\lambda,N}$ *(definition 2) and for all* p.p.t. $\mathcal{A}$

$$\left|\Pr\left[1 = \mathsf{Exp}^{\mathsf{PI\text{-}PRF}}_{N,\mathcal{A},f,0}\right] - \Pr\left[1 = \mathsf{Exp}^{\mathsf{PI\text{-}PRF}}_{N,\mathcal{A},1}\right]\right| = \mathit{negl}(\lambda),$$

  where $\mathsf{Exp}^{\mathsf{PI\text{-}PRF}}_{N,\mathcal{A},f,0}$ *and* $\mathsf{Exp}^{\mathsf{PI\text{-}PRF}}_{N,\mathcal{A},1}$ *are defined as in Figure 1d. Alternatively, if* f *satisfies the above property for a fixed* Sam *(and not necessarily for arbitrary admissible sampler), we say that* f *is a* Sam-PI-PRF*.*

[8] However, the definition of [PS08] does not cover the full class of admissible samplers, since their definition requires the samples to be *independent*, while we also allow correlated samples. Correlation is actually very crucial to our application, since we will later use a PRF with a fixed random key as a sampler - PRF outputs will be correlated, since they share the key.

[9] A non-adaptive PRF can be cast as a synthesiser.

Experiment Strong, non-adaptive, weak, and pseudorandom-input PRFs

```
$\mathsf{Exp}^{\mathsf{sPRF}}_{\mathcal{A},f,0}$
$k \xleftarrow{\$} \{0,1\}^\lambda$
return $\mathcal{A}^{\mathsf{EVAL}}(1^\lambda)$

$\mathsf{EVAL}(x)$
assert $x \in \{0,1\}^\lambda$
$y \leftarrow f(k,x)$
return $y$
```

```
$\mathsf{Exp}^{\mathsf{sPRF}}_{\mathcal{A},1}$
return $\mathcal{A}^{\mathsf{EVAL}}(1^\lambda)$

$\mathsf{EVAL}(x)$
assert $x \in \{0,1\}^\lambda$
$y \xleftarrow{\$} \{0,1\}^\lambda$
return $y$
```

(a) Security experiments for (strong) PRFs.

```
$\mathsf{Exp}^{\mathsf{naPRF}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),f,0}$
$(\vec{x}, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_0(1^\lambda)$
$k \xleftarrow{\$} \{0,1\}^\lambda$
for $i = 1, \ldots, |\vec{x}|$
    $\vec{y}_i \leftarrow f(k, \vec{x}_i)$
return $\mathcal{A}_1(\mathsf{st}, \vec{y})$
```

```
$\mathsf{Exp}^{\mathsf{naPRF}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),1}$
$\vec{x}, \mathsf{st} \xleftarrow{\$} \mathcal{A}_0(1^\lambda)$
for $i = 1, \ldots, |\vec{x}|$
    $\vec{y}_i \xleftarrow{\$} \{0,1\}^\lambda$
return $\mathcal{A}_1(\mathsf{st}, \vec{y})$
```

(b) Security experiments for naPRFs.

```
$\mathsf{Exp}^{\mathsf{wPRF}}_{p,\mathcal{A},f,0}$
$\vec{x} \xleftarrow{\$} \{0,1\}^{N\cdot\lambda}$
$k \xleftarrow{\$} \{0,1\}^\lambda$
for $i = 1, \ldots, N$
    $\vec{y}_i \leftarrow f(k, \vec{x}_i)$
return $\mathcal{A}(\vec{x}, \vec{y})$
```

```
$\mathsf{Exp}^{\mathsf{wPRF}}_{p,\mathcal{A},1}$
$\vec{x} \xleftarrow{\$} \{0,1\}^{N\cdot\lambda}$
for $i = 1, \ldots, N$
    $\vec{y}_i \xleftarrow{\$} \{0,1\}^\lambda$
return $\mathcal{A}(\vec{x}, \vec{y})$
```

(c) Security experiments for wPRFs.

```
$\mathsf{Exp}^{\mathsf{PI\text{-}PRF}}_{N,\mathcal{A},f,0}$
$r \xleftarrow{\$} \{0,1\}^{\mathtt{inLen}(\mathsf{Sam}_{\lambda,N})}$
$\vec{x} \leftarrow \mathsf{Sam}_{\lambda,N}(r)$
$k \xleftarrow{\$} \{0,1\}^\lambda$
for $i = 1, \ldots, N$
    $\vec{y}_i \leftarrow f(k, \vec{x}_i)$
return $\mathcal{A}(\vec{x}, \vec{y}, r)$
```

```
$\mathsf{Exp}^{\mathsf{PI\text{-}PRF}}_{N,\mathcal{A},1}$
$r \xleftarrow{\$} \{0,1\}^{\mathtt{inLen}(\mathsf{Sam}_{\lambda,N})}$
$\vec{x} \leftarrow \mathsf{Sam}_{\lambda,N}(r)$
for $i = 1, \ldots, N$
    $\vec{y}_i \xleftarrow{\$} \{0,1\}^\lambda$
return $\mathcal{A}(\vec{x}, \vec{y}, r)$
```

(d) Security experiments for PI-PRFs. Difference to wPRF is highlighted.

Fig. 1: Security experiments for strong, non-adaptive, weak, and pseudorandom-input pseudorandom functions.

It immediately follows from the definitions, that any non-adaptive PRF is also a pseudorandom-input PRF (for *all* admissible samplers), and that any pseudorandom-input PRF is also a weak PRF. However it may be *a priori* unclear how much stronger than a weak PRF a pseudorandom-input PRF is. Fortunately, by adapting a result of Pietrzak and Sjödin [PS08], we are able to provide a conditional argument towards closing the gap between the two notions. Namely, if there is a wPRF that is *not* also a PI-PRF (for *all* admissible samplers), then we can build *infinitely-often key agreement* from the wPRF and an adversary breaking its "PI-PRF properties".

Theorem 4 (wPRF not PI-PRF implies io-KA, adapted from [PS08]). *Let* wPRF *be a weak PRF. If* wPRF *is* not *a pseudorandom-input PRF, then there exists an infinitely often two-party key-agreement protocol.*

Since this theorem is a straightforward adaptation of a known theorem from the literature, we simply state it here and instead refer the reader to Appendix B.1 for a self-contained proof of Theorem 4.

3.2 From PI-PRF to sPRF

We now provide our instantiation of the transformation from a PI-PRF to a sPRF, which can be broken down modularly as a transformation from a PI-PRF to a naPRF and a transformation from a naPRF to a sPRF.

Lemma 5 (PI-PRF $\circ$ naPRF is a naPRF, Informal). *Let* PI-PRF *be a pseudorandom-input PRF, and let* naPRF *be a non-adaptive PRF. Then* $f(k_{\mathsf{sPRF}}, x) := \mathsf{PI\text{-}PRF}(k_{\mathsf{sPRF}}, \mathsf{naPRF}(k_{\mathsf{naPRF}}, x))$ *is a "fully non-adaptive PRF in the CRS model" (whose security is only guaranteed for queries non-adaptively chosen* before *seeing the CRS generated as* $\mathsf{Setup}(1^\lambda)$: $\{k_{\mathsf{naPRF}} \xleftarrow{\$} \{0,1\}^\lambda$; *Return* $k_{\mathsf{naPRF}}\}$*).*

Proof. In order to prove this lemma, we need to provide some formalism for the notion of a "fully non-adaptive PRF in the CRS model". Syntactically, it is defined as a pair of algorithms $(\mathsf{fnaPRF}, \mathsf{Setup})$ where $\mathsf{Setup}(1^\lambda)$ generates an (unstructured) CRS, while fnaPRF has the same syntax as a non-adaptive PRF (but takes a CRS as an additional input). Security is very analogous to the definition of a non-adaptive PRF, but with instead the following security experiments $\mathsf{Exp}^{\mathsf{fnaPRF}}_{p,\mathcal{A},f,0}$ and $\mathsf{Exp}^{\mathsf{fnaPRF}}_{p,\mathcal{A},1}$ (differences with the experiments for non-adaptive PRFs Figure 1b are highlighted):

```
$\mathsf{Exp}^{\mathsf{fnaPRF}}_{p,\mathcal{A},(f,\mathsf{Setup}),0} = \mathsf{Hyb}_0(1^\lambda)$
$\vec{x}, \mathsf{st} \leftarrow\$ \mathcal{A}_0(1^\lambda)$
$\mathsf{CRS} \leftarrow\$ \mathsf{Setup}(1^\lambda)$
$k_{\mathsf{sPRF}} \leftarrow\$ \{0,1\}^\lambda$
for $i = 1, ..., |\vec{x}|$
    $\vec{y}_i \leftarrow f(k_{\mathsf{sPRF}}, \vec{x}_i)$
return $\mathcal{A}_1(\mathsf{CRS}, \mathsf{st}, \vec{y})$
```

```
$\mathsf{Exp}^{\mathsf{fnaPRF}}_{\mathsf{p},\mathcal{A},\mathsf{Setup},1} = \mathsf{Hyb}_4(1^\lambda)$
$\vec{x}, \mathsf{st} \leftarrow\$ \mathcal{A}_0(1^\lambda)$
$\mathsf{CRS} \leftarrow\$ \mathsf{Setup}(1^\lambda)$
for $i = 1, ..., |\vec{x}|$
    $\vec{y}_i \leftarrow\$ \{0,1\}^\lambda$
return $\mathcal{A}_1(\mathsf{CRS}, \mathsf{st}, \vec{y})$
```

Let us now show that the experiments $\mathsf{Exp}^{\mathsf{fnaPRF}}_{p,\mathcal{A},(f,\mathsf{Setup}),0}$ and $\mathsf{Exp}^{\mathsf{fnaPRF}}_{\mathsf{p},\mathcal{A},\mathsf{Setup},1}$, where (f, Setup) are defined as in lemma 5, are indistinguishable by considering the hybrids $\mathsf{Hyb}_1, \mathsf{Hyb}_2, \mathsf{Hyb}_3$ of Figure 2.

Code Equivalence

Security of PI-PRF

```
$\mathsf{Hyb}_1(\lambda)$
$\vec{x}, \mathsf{st} \leftarrow\$ \mathcal{A}_0(1^\lambda)$
$k_{\mathsf{naPRF}} \leftarrow\$ \{0,1\}^\lambda$
$k_{\mathsf{sPRF}} \leftarrow\$ \{0,1\}^\lambda$
for $i = 1, ..., |\vec{x}|$
    $\vec{z}_i \leftarrow \mathsf{naPRF}(k_{\mathsf{naPRF}}, \vec{x}_i)$
    $\vec{y}_i \leftarrow \mathsf{PI\text{-}PRF}(k_{\mathsf{sPRF}}, \vec{z}_i)$
return $\mathcal{A}_1(k_{\mathsf{naPRF}}, \mathsf{st}, \vec{y})$
```

```
$\mathsf{Hyb}_2(\lambda)$
$r \leftarrow\$ \{0,1\}^{\mathrm{inLen}(\mathsf{Sam}_{\lambda,|\vec{x}|})}$
$\vec{z} \leftarrow \mathsf{Sam}_{\lambda,|\vec{x}|}(r)$
$k_{\mathsf{sPRF}} \leftarrow\$ \{0,1\}^\lambda$
for $i = 1, ..., |\vec{z}|$
    $\vec{y}_i \leftarrow \mathsf{PI\text{-}PRF}(k_{\mathsf{sPRF}}, \vec{z}_i)$
return $\mathcal{A}(r,\vec{y})$

$\mathsf{Sam}_{\lambda,|\vec{x}|}(r)$
$\mathsf{st}||k_{\mathsf{naPRF}} \leftarrow r$
$\vec{x}, \mathsf{st} \leftarrow \mathcal{A}_0(1^\lambda; \mathsf{st})$
for $i = 1, ..., |\vec{x}|$
    $\vec{z}_i \leftarrow \mathsf{naPRF}(k_{\mathsf{naPRF}}, \vec{x}_i)$
return $\vec{z}$
```

```
$\mathsf{Hyb}_3(\lambda)$
$r \leftarrow\$ \{0,1\}^{\mathrm{inLen}(\mathsf{Sam}_{\lambda,|\vec{x}|})}$
$\vec{z} \leftarrow \mathsf{Sam}_{\lambda,|\vec{x}|}(r)$
$k_{\mathsf{sPRF}} \leftarrow\$ \{0,1\}^\lambda$
for $i = 1, ..., |\vec{x}|$
    $\vec{y}_i \leftarrow\$ \{0,1\}^\lambda$
return $\mathcal{A}_1(r, \vec{y})$

$\mathsf{Sam}_{\lambda,|\vec{x}|}(r)$
$\mathsf{st}||k_{\mathsf{naPRF}} \leftarrow r$
$\vec{x}, \mathsf{st} \leftarrow \mathcal{A}_0(1^\lambda; \mathsf{st})$
for $i = 1, ..., |\vec{x}|$
    $\vec{z}_i \leftarrow \mathsf{naPRF}(k_{\mathsf{naPRF}}, \vec{x}_i)$
return $\vec{z}$
```

Fig. 2: Sequence of hybrids for proving *naPRF security* in the proof of Lemma 5.

- $\mathsf{Hyb}_0 \equiv \mathsf{Hyb}_1$: These hybrids are code-equivalent; we obtain Hyb_1 by simply inlining the definition of f (and of its setup $k_{\mathsf{naPRF}} \leftarrow\$ \{0,1\}^\lambda$).
- $\mathsf{Hyb}_1 \equiv \mathsf{Hyb}_2$: Again, these hybrids are in fact code-equivalent. Indeed, *w.l.o.g.* the second stage adversary's state st is equal to the first stage adversary's internal randomness. Thus, nothing changes if we define $r := \mathsf{st}||k_{\mathsf{naPRF}}$.
- $\mathsf{Hyb}_2 \overset{c}{\approx} \mathsf{Hyb}_3$: By *naPRF security* of naPRF, the sampler $\mathsf{Sam}_{\lambda,|\vec{x}|}$ is admissible. The hybrids are therefore indistinguishable by *PI-PRF security* of $\mathsf{PI\text{-}PRF}$.
- $\mathsf{Hyb}_3 \equiv \mathsf{Hyb}_4$: These hybrids are code-equivalent (by combining the same arguments used to show that $\mathsf{Hyb}_0 \equiv \mathsf{Hyb}_1$ and $\mathsf{Hyb}_1 \equiv \mathsf{Hyb}_2$, except in reverse).

□

Lemma 6 (naPRF ∘ ELF is a sPRF, Informal). *Let f be a "fully non-adaptive PRF in the CRS model" and let* ELF *be an extremely lossy function. Then* $\mathsf{sPRF}(k_{\mathsf{sPRF}}, x) := f(k_{\mathsf{sPRF}}, \mathsf{elf}(x))$ *is a strong PRF (in the CRS model), where the coins used to generate* $\mathsf{elf} \leftarrow\!\!\$\ \mathsf{ELF.Gen}(1^\lambda, 2^\lambda)$ *and coins needed for the fully non-adaptive PRF r_f are in the public CRS (*i.e. $\mathsf{Setup}(1^\lambda) : \{r_{\mathsf{ELF}} \leftarrow\!\!\$\ \{0,1\}^\lambda; r_f \leftarrow\!\!\$\ ;$ *Return* $r_{\mathsf{ELF}}, r_f\}$ *and* $\mathsf{elf} \leftarrow\!\!\$\ \mathsf{ELF.Gen}(1^\lambda, 2^\lambda; r_{\mathsf{ELF}})$*).*

Proof. Again, in order to provide a meaningful proof we need to introduce some formalism describing a strong PRF in the CRS model. The security experiments, which are analogous to those of a strong PRF (in the plain model), are the following (differences with fig. 1a are highlighted):

```
$\mathsf{Exp}^{\mathsf{sPRF}}_{\mathcal{A},(\mathsf{sPRF},\mathsf{Setup}),0} = \mathsf{Hyb}_0$
$\mathsf{CRS} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$k_{\mathsf{sPRF}} \leftarrow\!\!\$\ \{0,1\}^\lambda$
return $\mathcal{A}^{\mathsf{EVAL}}(1^\lambda, \mathsf{CRS})$

$\mathsf{EVAL}(x)$
assert $x \in \{0,1\}^\lambda$
$y \leftarrow \mathsf{sPRF}_{\mathsf{CRS}}(k_{\mathsf{sPRF}}, x)$
return $y$
```

```
$\mathsf{Exp}^{\mathsf{sPRF}}_{\mathcal{A},\mathsf{Setup},1} = \mathsf{Hyb}_5$
$\mathsf{CRS} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$

return $\mathcal{A}^{\mathsf{EVAL}}(1^\lambda, \mathsf{CRS})$

$\mathsf{EVAL}(x)$
assert $x \in \{0,1\}^\lambda$
$y \leftarrow\!\!\$\ \{0,1\}^\lambda$
return $y$
```

We now show via several game hops that the experiments $\mathsf{Exp}^{\mathsf{sPRF}}_{\mathcal{A},\mathsf{sPRF},0}$ and $\mathsf{Exp}^{\mathsf{sPRF}}_{\mathcal{A},1}$ (as parameterised by the pair $(\mathsf{sPRF}, \mathsf{Setup})$ defined in lemma 6) are indistinguishable. Suppose for contradiction, that a *p.p.t.* adversary $\mathcal{A}$ has non-negligible advantage in distinguishing the PRF games of sPRF. Let r be a sufficiently large polynomial such that $\mathcal{A}$ cannot distinguish an ELF with image size r from an injective ELF.

Security of ELF: $\mathsf{Hyb}_1 \to \mathsf{Hyb}_2$ Code Equivalence: $\mathsf{Hyb}_2 \to \mathsf{Hyb}_3$ fnaPRF Security of f: $\mathsf{Hyb}_3 \to \mathsf{Hyb}_4$

```
$\mathsf{Hyb}_1(\lambda)$
$\mathsf{elf} \leftarrow\!\!\$\ \mathsf{ELF.Gen}(1^\lambda, 2^\lambda)$
$k_{\mathsf{sPRF}} \leftarrow\!\!\$\ \{0,1\}^\lambda$
return $\mathcal{A}^{\mathsf{EVAL}}(1^\lambda, \mathsf{elf}, r_f)$

$\mathsf{EVAL}(x)$
assert $x \in \{0,1\}^\lambda$
$y \leftarrow f(k_{\mathsf{sPRF}}, \mathsf{elf}(x))$
return $y$
```

```
$\mathsf{Hyb}_2(\lambda)$
$\mathsf{elf} \leftarrow\!\!\$\ \mathsf{ELF.Gen}(M, r)$
$k_{\mathsf{sPRF}} \leftarrow\!\!\$\ \{0,1\}^\lambda$
return $\mathcal{A}^{\mathsf{EVAL}}(1^\lambda, \mathsf{elf}, r_f)$

$\mathsf{EVAL}(x)$
assert $x \in \{0,1\}^\lambda$
$y \leftarrow f(k_{\mathsf{sPRF}}, \mathsf{elf}(x))$
return $y$
```

```
$\mathsf{Hyb}_3(\lambda)$
$\mathsf{elf} \leftarrow\!\!\$\ \mathsf{ELF.Gen}(1^\lambda, r)$
$k_{\mathsf{sPRF}} \leftarrow\!\!\$\ \{0,1\}^\lambda$
for $z \in \mathrm{Im}(\mathsf{elf})$ :
  $T[z] \leftarrow f(k_{\mathsf{sPRF}}, z)$
return $\mathcal{A}^{\mathsf{EVAL}}(1^\lambda, \mathsf{elf}, r_f)$

$\mathsf{EVAL}(x)$
assert $x \in \{0,1\}^\lambda$
$y \leftarrow T[\mathsf{elf}(x)]$
return $y$
```

```
$\mathsf{Hyb}_4(\lambda)$
$\mathsf{elf} \leftarrow\!\!\$\ \mathsf{ELF.Gen}(1^\lambda, r)$
$k_{\mathsf{sPRF}} \leftarrow\!\!\$\ \{0,1\}^\lambda$
for $z \in \mathrm{Im}(\mathsf{elf})$ :
  $T[z] \leftarrow\!\!\$\ \{0,1\}^\lambda$
return $\mathcal{A}^{\mathsf{EVAL}}(1^\lambda, \mathsf{elf}, r_f)$

$\mathsf{EVAL}(x)$
assert $x \in \{0,1\}^\lambda$
$y \leftarrow T[\mathsf{elf}(x)]$
return $y$
```

Fig. 3: Sequence of hybrids for proving *sPRF security* in the proof of Lemma 6.

- $\mathsf{Hyb}_0 \equiv \mathsf{Hyb}_1$: These hybrids are code-equivalent, we simply inlined the definitions of sPRF and Setup
- $\mathsf{Hyb}_1 \overset{c}{\approx} \mathsf{Hyb}_2$: These hybrids are indistinguishable by applying the security of ELF.
- $\mathsf{Hyb}_2 \equiv \mathsf{Hyb}_3$: These hybrids are code-equivalent; the only difference is pre-processing oracle calls by generating a lookup table.

- $\mathsf{Hyb}_3 \overset{c}{\approx} \mathsf{Hyb}_4$: These hybrids are indistinguishable by *fully non-adaptive PRF security (in the CRS model)* of f. More precisely, in the fnaPRF experiment, the first stage adversary $\mathcal{A}_0^f$ samples the ELF and chooses the image of the ELF as the vector $\vec{x}$ and the description of the ELF elf as the state $\mathsf{st} = \mathsf{elf}$ that is passed to the second stage adversary, in this case $\mathcal{A}_1^f = \mathcal{A}^{\mathsf{EVAL}}$. Now, since an arbitrary PPT $\mathcal{A}_1^f$ cannot distinguish the naPRF f outputs from random, neither can $\mathcal{A}^{\mathsf{EVAL}}$ who must run in time $<< r$ (note that arbitrary PPT adversary $\mathcal{A}_1^f$ can emulate the EVAL oracle calls by computing the full ELF image of size r).
- $\mathsf{Hyb}_4 \overset{c}{\approx} \mathsf{Hyb}_5$: The hybrids are equivalent, by *ELF security* of ELF, with the observation we applied the reverse of the code-equivalent transform between games Hyb_0 and Hyb_1.

□

By combining Lemmas 5 and 6 we immediately obtain Corollary 7.

Corollary 7 (PI-PRF + naPRF + ELF is PRF, Informal). *Let* $\mathsf{PI\text{-}PRF}$ *be a pseudorandom-input PRF, let* naPRF *be a non-adaptive PRF, and let* ELF *be an extremely lossy function. Then*

$$\mathsf{sPRF}(k_{\mathsf{sPRF}}, x) := \mathsf{PI\text{-}PRF}(k_{\mathsf{sPRF}}, \underbrace{\mathsf{naPRF}(k_{\mathsf{naPRF}}, \mathsf{elf}(x))}_{\textit{public pre-processing}})$$

is a strong PRF (in the CRS model), where $k_{\mathsf{naPRF}} \leftarrow\!\!\$ \; \{0,1\}^\lambda$ *and the coins used to generate* $\mathsf{elf} \leftarrow\!\!\$$ $\mathsf{ELF.Gen}(1^\lambda, 2^\lambda)$ *are in a public CRS.*

3.3 Direct Implications for Oblivious PRFs

We established in corollary 7 that the random oracle classically used to transform a weak PRF to a strong one can be instantiated, provided we are willing to assume the weak PRF is in fact a *pseudorandom-input PRF*. While we would argue this is already of theoretical interest, one may pause and wonder why one would ever use this transformation given that a strong PRF can be built in a black-box way from a weak PRF, and *a fortiori* from a pseudorandom-input PRF. Where the hash-and-evaluate paradigm truly shines is in the distributed setting, where it allows us to only wrap the compiler, that is *secure multiparty computation*, around a (low-complexity) weak PRF. This idea of applying a random oracle to the input before performing the secure evaluation of only a *weak* PRF is not merely of theoretical interest, but rather is a key ingredient in the state-of-the-art OPRF of Dinur et al. [DGH+21].

An *Oblivious PRF* (OPRF) is a secure two-party protocol realising the functionality $(k, x) \mapsto (\perp, F(k, x))$ for some pseudorandom function family F. If F is no longer assumed to be a strong PRF but instead only a weak or pseudorandom-input PRF, we will call such a protocol a *secure function evaluation (SFE) of a weak (*resp. *pseudorandom-input) PRF.* We refer to remark 8 for a discussion on why we do not use the terms *Oblivious weak/pseudorandom-input PRF.*

Remark 8 (Defining an "Oblivious wPRF"). The problem of defining an "Oblivious weak PRF"[10] is a delicate one, which was explicitly left open by *e.g.* [JKR19, CHL22b]. A first attempt would be to define it as *Secure Function Evaluation (SFE) of a weak PRF*, *i.e.* as a secure two-party protocol realising the functionality $(k, x) \mapsto (\perp, F(k, x))$ for some *weak* pseudorandom function family F. This is a convenient solution from a design perspective, but it places the burden of not misusing the primitive on the user (wishing to build some larger protocol). Indeed, using such a protocol only guarantees server privacy *over the randomness of the queries* made by the client. When the primitive of *SFE of a wPRF* is composed, it becomes unclear what this means[11]; in particular, in Canetti's *Universal Composability* framework [Can01] the inputs of even semi-honest parties are assumed to have been provided by a malicious environment, so even "trusting a semi-honest party to use random

[10] Not to be confused with a *weak OPRF*, *a.k.a.* a *relaxed OPRF*, which is a relaxation of an OPRF introduced by Freedman et al. [FIPR05] which allows for some leakage of the key to the client.

[11] Say a client and a server run tho parallel instances of SFE of a wPRF, and the client queries an random input x in the first instance, and $x + 1$ in the next: the inputs used by the client in each instance are random, but nevertheless correlated, and server security is not expected to hold.

inputs” is not necessarily sound, unless the protocol explicitly specifies how they should be sampled. For this reason, one might argue that the ideal functionality of an *Oblivious weak PRF* should sample the queries itself, and *output* them to the client, alongside their evaluations. This definition would be analogous to those of *random OT* [Rab05] and *random-input PIR* [GHM$^+$21]. The downside of this alternative definition is that it does not seem possible to then use the hash-then-evaluate paradigm to boost an oblivious wPRF to an OPRF.

Lemma 9 (Oblivious Hash-then-Evaluate PRF). *Let* $\mathsf{PI\text{-}PRF}\colon \{0,1\}^{\lambda} \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}$ *be a pseudorandom-input PRF, let* $\mathsf{ELF.Gen}$ *be an extremely lossy function, and let* $\mathsf{naPRF}\colon \{0,1\}^{\lambda} \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{n(\lambda)}$ *be a non-adaptive PRF. Then the protocol of Figure 4 (defined in the* $\mathcal{F}_{\mathsf{SFE}}(\mathsf{PI\text{-}PRF})$*-hybrid model, where* $\mathcal{F}_{\mathsf{SFE}}(\mathsf{PI\text{-}PRF}(\cdot,\cdot))$ *is the ideal functionality computing* $(k,x) \mapsto (\bot, \mathsf{wPRF}(k,x))$*) is a (semi-honest) OPRF in the CRS model for the following PRF:*

$$\begin{array}{lrl} \mathsf{PRF}\colon & \{0,1\}^{3\lambda} \times \{0,1\}^{n(\lambda)} & \to \{0,1\}^{m(\lambda)} \\ & (k = (k_{\mathsf{PI\text{-}PRF}}, k_{\mathsf{naPRF}}, r), x) & \mapsto \mathsf{PI\text{-}PRF}(k_{\mathsf{PI\text{-}PRF}}, \mathsf{naPRF}(k_{\mathsf{naPRF}}, f(x))), \\ & & \quad \textit{where } f = \mathsf{ELF.Gen}(2^{n(\lambda)}, 2^{n(\lambda)}; r) \end{array}$$

Proof Sketch. There are two statements to prove: the first is that PRF is a pseudorandom function family, and the second is that the fig. 4 securely realises the functionality $(k,x) \mapsto (\bot, \mathsf{PRF}(k,x))$. We already proved the former in corollary 7, and the latter follows immediately from the fact the only interaction between C and S is through $\mathcal{F}_{\mathsf{SFE}}(\mathsf{PI\text{-}PRF}(\cdot,\cdot))$. □

Protocol Π_{OPRF}

Parties: C (the client) and S (the server)

Parameters: $\mathsf{PI\text{-}PRF}(\cdot,\cdot)\colon \{0,1\}^{\lambda} \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}$ is a pseudorandom-input PRF, $\mathsf{naPRF}(\cdot,\cdot)\colon \{0,1\}^{\lambda} \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{n(\lambda)}$ is a non-adaptive PRF, and $\mathsf{ELF.Gen}$ is an ELF.

Hybrid Model: The protocol is defined in the $\mathcal{F}_{\mathsf{SFE}}(\mathsf{PI\text{-}PRF}(\cdot,\cdot))$-hybrid model.

Input: S holds as input a PI-PRF key $k_{\mathsf{PI\text{-}PRF}} \in \{0,1\}^{\lambda}$, a naPRF key $k_{\mathsf{naPRF}} \in \{0,1\}^{\lambda}$, and randomness $r \in \{0,1\}^{\lambda}$; and C holds as input $x \in \{0,1\}^{n(\lambda)}$.

Setup: The CRS is a 2λ-bit random string.

The Protocol:

1. C parses the CRS as (k_{naPRF}, r), where $k_{\mathsf{naPRF}}, r \in \{0,1\}^{\lambda}$
2. C computes computes $f \leftarrow \mathsf{ELF.Gen}(2^{n(\lambda)}, 2^{n(\lambda)}; r)$
3. S and C send respectively $(\texttt{server}, k_{\mathsf{PI\text{-}PRF}})$ and $(\texttt{client}, \mathsf{naPRF}(f(x))$ to $\mathcal{F}_{\mathsf{SFE}}(\mathsf{PI\text{-}PRF}(\cdot,\cdot))$, and C waits to receive $y \in \{0,1\}^{m(\lambda)}$ from $\mathcal{F}_{\mathsf{SFE}}(\mathsf{PI\text{-}PRF}(\cdot,\cdot))$.
4. S outputs $\bot$, and C outputs y.

Fig. 4: OPRF (parameterised by the PRF of corollary 7) given secure function evaluation of a pseudorandom-input PRF.

Remark 10 (Instantiating Sate-of-the-Art OPRF). We recall that the OPRF construction of Dinur et al. [DGH$^+$21], using only two rounds and 641 bits of online communication, boils down to providing a special-purpose protocol for securely computing Boneh et al. s [BIP$^+$18] weak PRF candidate. Under the assumption that this candidate is in fact a pseudorandom-input PRF (for some class of admissible samplers)—we discuss this assumption in section 5.1—then the construction of fig. 4 can be used to instantiate Dinur et al.’s [DGH$^+$21] OPRF while preserving the number of rounds and the amount of communication. Depending on the desired level of security (*e.g.* malicious), some additional tools will be required.

4 Instantiating Hash-then-Evaluate: From PI-PCF to PCF

Section 3.2 shows that the random oracle which boosts a weak PRF to a strong PRF—up to the pseudorandom-input caveat—can be instantiated. Seeing *function secret sharing (FSS)* as a compiler which turns a (weak, non-adaptive, strong) PRF into a (weak, non-adaptive, strong) PCF (cf. [BCG$^+$20, Theorem 5.5]), the results in Section 3.2 have direct implications to PCFs. In this section, we show that our weak-to-non-adaptive-to-strong transforms can in fact be applied to *any* PCF (without having to assume it was obtained by using FSS).

Section 4.1 recalls the existing notions of PCF from [BCG$^+$20]: weak (wPCF), non-adaptive (naPCF), and strong PCFs (sPCF). Section 4.2 introduces the notions of *pseudorandom-input PCF (PI-PCF)* and *fully non-adaptive PCF (fnaPCF)*. Sections 4.3 and 4.4 introduce transforms to boost a PI-PCF to a fnaPCF, and a fnaPCF to a sPCF, respectively.

4.1 Existing flavours of Pseudorandom Correlation Functions

At a high level, a *pseudorandom correlation function* (PCF) cryptographically compresses (superpolynomial-size) correlated random strings from some ideal correlation, *e.g.* generating long vectors of Beaver triples [Bea92][12], down to short keys. Given a key, it should be possible to incrementally recover parts of the long string, *e.g.* evaluating the PCF key at position i should yield a party's share of the i^{th} Beaver triple. Prior works have considered three different flavours of PCFs, from weakest to strongest: *weak* PCFs (wPCF), *non-adaptive* PCFs (naPCF), and *strong* PCFs (sPCF). Intuitively, and analogously to their PRF counterparts, security is guaranteed (*e.g.* the pseudorandom Beaver triples are "safe to use") when evaluating the PCF keys at random (resp. non-adaptively chosen, resp. any) points. Note that contrary to PRFs, the PCF literature treats *weak* PCFs (security w.r.t. random inputs) as the default PCF which is motivated in part by [BCG$^+$20, Theorem 4.5], which shows that the hash-then-evaluate paradigm can be used to turn a weak PCF into a strong one.

For technical reasons, and in order to provide a meaningful definition of PCF for infinite families of finite correlations, we only consider *reverse sampleable* correlations (Definition 11). We refer to [BCG$^+$20, Section 4] for more details.

Definition 11 (Reverse-Sampleable Correlation, [BCG$^+$19]). *Let* $1 \leq \ell_0(\lambda), \ell_1(\lambda) \leq \mathsf{poly}(\lambda)$ *be output-length functions. Let* $\mathcal{Y}$ *be a probabilistic algorithm on input* 1^λ*, returns a pair of outputs* $(y_0, y_1) \in \{0,1\}^{\ell_0(\lambda)} \times \{0,1\}^{\ell_1(\lambda)}$*, defining a correlation on the outputs.*
We say that $\mathcal{Y}$ *defines a* reverse-sampleable *correlation if there exists a PPT algorithm* $\mathsf{RSample}$ *which takes as input* 1^λ*,* $\sigma \in \{0,1\}$*, and* $y_\sigma \in \{0,1\}^{\ell_\sigma(\lambda)}$*, and outputs* $y_{1-\lambda}^{\ell_{1-\sigma}(\lambda)}$*, such that for all* $\sigma \in \{0,1\}$

$$\{(y_0, y_1) \colon (y_0, y_1) \leftarrow\!\!\$\, \mathcal{Y}(1^\lambda)\} \text{ and } \{(y_0, y_1) \colon (y'_0, y'_1) \leftarrow\!\!\$\, \mathcal{Y}(1^\lambda), y_\sigma \leftarrow y'_\sigma, y_{1-\sigma} \leftarrow \mathsf{RSample}(1^\lambda, \sigma, y_\sigma)\} \ .$$

are statistically close.

All the different flavours of PCF admit the same syntax, which we describe in Definition 12.

Definition 12 (Pseudorandom Correlation Function – Syntax [BCG$^+$20, Definition 4.3]). *Let* $\mathcal{Y}$ *be a reverse-sampleable correlation with output length functions* $\ell_0(\lambda), \ell_1(\lambda)$ *and let* $\lambda \leq n(\lambda) \leq \mathsf{poly}(\lambda)$ *be an input length function. Syntactically, a* pseudorandom correlation generator *is a pair of algorithms* $\mathsf{PCF} = (\mathsf{PCF.Gen}, \mathsf{PCF.Eval})$ *with the following syntax:*

- $\mathsf{wPCF.Gen}(1^\lambda)$ *is a probabilistic polynomial time algorithm that on input* 1^λ*, outputs a pair of keys* (k_0, k_1)*; we assume that* λ *can be inferred from the keys.*
- $\mathsf{wPCF.Eval}(\sigma, k_\sigma, x)$ *is a deterministic polynomial time algorithm that on input* $\sigma \in \{0,1\}$*, key* k_σ *and input value* $x \in \{0,1\}^{n(\lambda)}$*, outputs a value* $y_\sigma \in \{0,1\}^{\ell_\sigma(\lambda)}$*.*

[12] Recall that multiplication triples are linear shares $[a], [b], [c]$ of some random multiplication triple ($a, b, c = ab$) where $a, b \leftarrow\!\!\$\, \mathcal{R}$ where $\mathcal{R}$ is some ring. As shown by Beaver [Bea92] parties holding linear shares of two different inputs $x, y \in \mathcal{R}$ can compute linear shares of $x \cdot y$ by: (1) locally computing shares of $\alpha = x - a$ and $\beta = y - b$ as $[\alpha] \leftarrow [x] - [a]$ and $[\beta] \leftarrow [y] - [b]$, (2) broadcasting the shares of α and β to reconstruct these values, (3) locally setting $[x \cdot y] \leftarrow \alpha \cdot [y] + \beta \cdot [x] - \alpha \cdot \beta + [c]$.

4.1.1 (Weak) Pseudorandom Correlation Function (wPCF). A PCF (with the syntax of Definition 12) is said to be a secure *weak pseudorandom correlation function (wPCF)* if it satisfies the properties of Definitions 13 and 14. At a high level, the property of *(weak) pseudorandom $\mathcal{Y}$-correlated outputs* states that the evaluations of the PCF (on truly random points) should look like samples from the ideal distribution $\mathcal{Y}$ *from the point of view of an external adversary (who does not hold a PCF key).* The *(weak) PCF security* property captures that a player holding a PCF key and seeing the other PCF key's evaluation at random points should learn "nothing about the other PCF key, except for its evaluation at those points".

Definition 13 ((Weakly) pseudorandom $\mathcal{Y}$-correlated outputs of a PCF). *For every non-uniform adversary $\mathcal{A}$ of size $B(\lambda)$, it holds that for all sufficiently large λ,*

$$\left|\Pr[\mathsf{Exp}^{\mathsf{w\text{-}pr}}_{\mathcal{A},N,0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{w\text{-}pr}}_{\mathcal{A},N,1}(\lambda) = 1]\right| \leq \epsilon(\lambda)$$

where $\mathsf{Exp}^{\mathsf{w\text{-}pr}}_{\mathcal{A},N,b}$ ($b \in \{0,1\}$) is defined as in Figure 5. In particular, the adversary is given access to $N(\lambda)$ samples.

Experiment (Weakly) Pseudorandom Correlated Outputs

```
$\mathsf{Exp}^{\mathsf{w\text{-}pr}}_{\mathcal{A},N,0}(\lambda)$

for $i = 1 \ldots N(\lambda)$ :
    $x^{(i)} \leftarrow\!\!\$ \{0,1\}^{n(\lambda)}$
    $(y_0^{(i)}, y_1^{(i)}) \hookleftarrow \mathcal{Y}(1^\lambda)$
$b \leftarrow \mathcal{A}(1^\lambda, (x^{(i)}, y_0^{(i)}, y_1^{(i)})_{i \in [N(\lambda)]})$
return $b$
```

```
$\mathsf{Exp}^{\mathsf{w\text{-}pr}}_{\mathcal{A},N,1}(\lambda)$
$(k_0, k_1) \leftarrow\!\!\$ \mathsf{wPCF.Gen}(1^\lambda)$
for $i = 1 \ldots N(\lambda)$ :
    $x^{(i)} \leftarrow\!\!\$ \{0,1\}^{n(\lambda)}$
    for $\sigma \in \{0,1\}$ :
        $y_\sigma^{(i)} \leftarrow\!\!\$ \mathsf{wPCF.Eval}(\sigma, k_\sigma, x^{(i)})$
$b \leftarrow \mathcal{A}(1^\lambda, (x^{(i)}, y_0^{(i)}, y_1^{(i)})_{i \in [N(\lambda)]})$
return $b$
```

Fig. 5: (Weakly) Pseudorandom $\mathcal{Y}$-correlated outputs of a wPCF.

Definition 14 ((Weak) PCF Security). *For every $\sigma \in \{0,1\}$ and every non-uniform adversary $\mathcal{A}$ of size $B(\lambda)$, it holds that for all sufficiently large λ,*

$$|\Pr[\mathsf{Exp}^{\mathsf{w\text{-}sec}}_{\mathcal{A},N,\sigma,0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{w\text{-}sec}}_{\mathcal{A},N,\sigma,1}(\lambda) = 1]| \leq \epsilon(\lambda)$$

where $\mathsf{Exp}^{\mathsf{w\text{-}sec}}_{\mathcal{A},N,\sigma,b}$ ($b \in \{0,1\}$) is defined as in Figure 6. In particular, the adversary is given access to $N(\lambda)$ samples (or simply N if there is no ambiguity).

Experiment (Weak) PCF Security

```
$\mathsf{Exp}^{\mathsf{w\text{-}sec}}_{\mathcal{A},N,\sigma,0}(\lambda)$
$(k_0, k_1) \leftarrow\!\!\$ \mathsf{wPCF.Gen}(1^\lambda)$
for $i = 1 \ldots N(\lambda)$ :
    $x^{(i)} \leftarrow\!\!\$ \{0,1\}^{n(\lambda)}$
    $y_{1-\sigma}^{(i)} \leftarrow\!\!\$ \mathsf{wPCF.Eval}(1-\sigma, k_{1-\sigma}, x^{(i)})$
$b \leftarrow \mathcal{A}(1^\lambda, \sigma, k_\sigma, (x^{(i)}, y_{1-\sigma}^{(i)})_{i \in [N(\lambda)]})$
return $b$
```

```
$\mathsf{Exp}^{\mathsf{w\text{-}sec}}_{\mathcal{A},N,\sigma,1}(\lambda)$
$(k_0, k_1) \leftarrow\!\!\$ \mathsf{wPCF.Gen}(1^\lambda)$
for $i = 1 \ldots N(\lambda)$ :
    $x^{(i)} \leftarrow\!\!\$ \{0,1\}^{n(\lambda)}$
    $y_\sigma^{(i)} \leftarrow\!\!\$ \mathsf{wPCF.Eval}(\sigma, k_\sigma, x^{(i)})$
    $y_{1-\sigma}^{(i)} \leftarrow\!\!\$ \mathsf{RSample}(1^\lambda, \sigma, y_\sigma^{(i)})$
$b \leftarrow \mathcal{A}(1^\lambda, \sigma, k_\sigma, (x^{(i)}, y_{1-\sigma}^{(i)})_{i \in [N(\lambda)]})$
return $b$
```

Fig. 6: Security of a wPCF. $\mathsf{RSample}$ is the algorithm for reverse sampling $\mathcal{Y}$ as in Definition 11.

4.1.2 Non-Adaptive Pseudorandom Correlation Function (naPCF). A PCF $\mathsf{naPCF} = (\mathsf{naPCF.Gen}, \mathsf{naPCF.Eval})$ (with the syntax of Definition 12) is said to be a secure *non-adaptive pseudorandom correlation function (naPCF)* if it satisfies the properties of Definitions 15 and 16. These properties are analogous to the weak counterpart, but hold on non-adaptively chosen queries, instead of only on truly random ones.

Definition 15 (Non-adaptively pseudorandom $\mathcal{Y}$-correlated outputs). *For non-uniform adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ of size $B(\lambda)$ asking at most $N(\lambda)$ non-adaptive queries to the oracle $\mathcal{O}_b(\cdot)$ (as defined in Figure 7), it holds that for all sufficiently large λ,*

$$\left|\Pr[\mathsf{Exp}^{\mathsf{na\text{-}pr}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),N,0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{na\text{-}pr}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),N,1}(\lambda) = 1]\right| \leq \epsilon(\lambda)$$

where $\mathsf{Exp}^{\mathsf{na\text{-}pr}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),N,b}$ ($b \in \{0,1\}$) is defined as in Figure 7.

Experiment Non-Adaptively Pseudorandom Correlated Outputs

```
$\mathsf{Exp}^{\mathsf{na\text{-}pr}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),N,0}(\lambda)$
$((x^{(i)})_{i\in[N(\lambda)]}, \mathsf{state}) \leftarrow\$ \mathcal{A}_0(1^\lambda)$
for $i = 1$ to $N(\lambda)$ :
    $(y_0^{(i)}, y_1^{(i)}) \leftarrow\$ \mathcal{Y}$
$b \leftarrow\$ \mathcal{A}_1(\mathsf{state}, (y_0^{(i)}, y_1^{(i)})_{i\in[N(\lambda)]})$
return $b$
```

```
$\mathsf{Exp}^{\mathsf{na\text{-}pr}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),N,1}(\lambda)$
$((x^{(i)})_{i\in[N(\lambda)]}, \mathsf{state}) \leftarrow\$ \mathcal{A}_0(1^\lambda)$
$(k_0, k_1) \leftarrow\$ \mathsf{naPCF.Gen}(1^\lambda)$
for $i = 1$ to $N(\lambda)$ :
  for $\sigma \in \{0,1\}$ :
    $y_\sigma^{(i)} \leftarrow \mathsf{naPCF.Eval}(\sigma, k_\sigma, x^{(i)})$
$b \leftarrow\$ \mathcal{A}_1(\mathsf{state}, (y_0^{(i)}, y_1^{(i)})_{i\in[N(\lambda)]})$
return $b$
```

Fig. 7: Non-Adaptively Pseudorandom $\mathcal{Y}$-correlated outputs of a naPCF.

Definition 16 (Non-Adaptive PCF Security). *For every $\sigma \in \{0,1\}$ and every non-uniform adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ of size $B(\lambda)$, it holds that for all sufficiently large λ,*

$$\left|\Pr[\mathsf{Exp}^{\mathsf{na\text{-}sec}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),N,\sigma,0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{na\text{-}sec}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),N,\sigma,1}(\lambda) = 1]\right| \leq \epsilon(\lambda)$$

where $\mathsf{Exp}^{\mathsf{na\text{-}sec}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),N,\sigma,b}$ ($b \in \{0,1\}$) is defined as in Figure 8.

Experiment Non-Adaptive PCF Security

```
$\mathsf{Exp}^{\mathsf{na\text{-}sec}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),N,\sigma,0}(\lambda)$
$(k_0, k_1) \leftarrow\$ \mathsf{naPCF.Gen}(1^\lambda)$
$((x^{(i)})_{i\in[N(\lambda)]}, \mathsf{state}) \leftarrow\$ \mathcal{A}_0(1^\lambda, \sigma, k_\sigma)$
for $i = 1$ to $N(\lambda)$ :
    $y_{1-\sigma}^{(i)} \leftarrow \mathsf{naPCF.Eval}(1-\sigma, k_{1-\sigma}, x^{(i)})$
$b \leftarrow\$ \mathcal{A}_1(1^\lambda, \sigma, \mathsf{state}, k_\sigma, (x^{(i)}, y_{1-\sigma}^{(i)})_{i\in[N(\lambda)]})$
return $b$
```

```
$\mathsf{Exp}^{\mathsf{na\text{-}sec}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),N,\sigma,1}(\lambda)$
$(k_0, k_1) \leftarrow\$ \mathsf{naPCF.Gen}(1^\lambda)$
$((x^{(i)})_{i\in[N(\lambda)]}, \mathsf{state}) \leftarrow\$ \mathcal{A}_0(1^\lambda, \sigma, k_\sigma)$
for $i = 1$ to $N(\lambda)$ :
    $y_\sigma^{(i)} \leftarrow \mathsf{naPCF.Eval}(\sigma, k_\sigma, x^{(i)})$
    $y_{1-\sigma}^{(i)} \leftarrow\$ \mathsf{RSample}(1^\lambda, \sigma, y_\sigma^{(i)})$
$b \leftarrow\$ \mathcal{A}_1(1^\lambda, \sigma, \mathsf{state}, k_\sigma, (x^{(i)}, y_{1-\sigma}^{(i)})_{i\in[N(\lambda)]})$
return $b$
```

Fig. 8: Security of a non-adaptive PCF. Here, $\mathsf{RSample}$ is the algorithm for reverse sampling $\mathcal{Y}$ as in Definition 11.

4.1.3 Strong Pseudorandom Correlation Function (sPCF). A PCF $\mathsf{sPCF} = (\mathsf{sPCF.Gen}, \mathsf{sPCF.Eval})$ (with the syntax of Definition 12) is said to be a secure *strong pseudorandom correlation function (sPCF)* if it satisfies the properties of Definitions 17 and 18. These properties are analogous to the non-adaptive counterpart, but hold on adaptively chosen queries, instead of only on non-adaptive ones.

Definition 17 (Strongly pseudorandom $\mathcal{Y}$-correlated outputs). *For every non-uniform adversary $\mathcal{A}$ of size $B(\lambda)$ asking at most $N(\lambda)$ queries to the oracle $\mathcal{O}_b(\cdot)$ (as defined in Figure 9), it holds that for all sufficiently large λ,*

$$\left|\Pr[\mathsf{Exp}^{\mathsf{s\text{-}pr}}_{\mathcal{A},0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{s\text{-}pr}}_{\mathcal{A},1}(\lambda) = 1]\right| \leq \epsilon(\lambda)$$

where $\mathsf{Exp}^{\mathsf{s\text{-}pr}}_{\mathcal{A},b}$ ($b \in \{0,1\}$) is defined as in Figure 9.

Experiment Strongly Pseudorandom Correlated Outputs

```
$\mathsf{Exp}^{\mathsf{s\text{-}pr}}_{\mathcal{A},b}(\lambda)$
$(k_0, k_1) \leftarrow\$ \mathsf{sPCF.Gen}(1^\lambda)$
$\mathcal{Q} \leftarrow \varnothing$
$b \leftarrow\$ \mathcal{A}^{\mathcal{O}_b(\cdot)}(1^\lambda)$
return $b$

$\mathcal{O}_0(x)$
if $(x, y_0, y_1) \in \mathcal{Q}$ :
    return $(y_0, y_1)$
else :
    $(y_0, y_1) \leftarrow\$ \mathcal{Y}(1^\lambda)$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(x, y_0, y_1)\}$
    return $(y_0, y_1)$

$\mathcal{O}_1(x)$
for $\sigma \in \{0,1\}$ :
    $y_\sigma \leftarrow \mathsf{sPCF.Eval}(1^\lambda, \sigma, k_\sigma, x)$
return $(y_0, y_1)$
```

Fig. 9: Strongly Pseudorandom $\mathcal{Y}$-correlated outputs of a sPCF.

Definition 18 (Strong PCF Security). *For every $\sigma \in \{0,1\}$ and every non-uniform adversary $\mathcal{A}$ of size $B(\lambda)$ asking at most $N(\lambda)$ queries to the oracle $\mathcal{O}_b(\cdot)$ (as defined in Figure 10), it holds that for all sufficiently large λ,*

$$|\Pr[\mathsf{Exp}^{\mathsf{s\text{-}sec}}_{\mathcal{A},0,\sigma}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{s\text{-}sec}}_{\mathcal{A},1,\sigma}(\lambda) = 1]| \leq \epsilon(\lambda)$$

where $\mathsf{Exp}^{\mathsf{s\text{-}sec}}_{\mathcal{A},\sigma}$ is defined as in Figure 10.

Experiment Strong PCF Security

```
$\mathsf{Exp}^{\mathsf{na\text{-}sec}}_{\mathcal{A},b,\sigma}(\lambda)$
$(k_0, k_1) \leftarrow\$ \mathsf{sPCF.Gen}(1^\lambda)$
$b \leftarrow\$ \mathcal{A}^{\mathcal{O}_b(\cdot)}(1^\lambda, \sigma, k_\sigma)$
return $b$

$\mathcal{O}_0(x)$
$y_{1-\sigma} \leftarrow \mathsf{sPCF.Eval}(1-\sigma, k_{1-\sigma}, x)$
return $y_{1-\sigma}$

$\mathcal{O}_1(x)$
$y_\sigma \leftarrow \mathsf{sPCF.Eval}(\sigma, k_\sigma, x)$
$y_{1-\sigma} \leftarrow \mathsf{RSample}(1^\lambda, \sigma, y_\sigma)$
return $y_{1-\sigma}$
```

Fig. 10: Security of a strong PCF. Here, $\mathsf{RSample}$ is the algorithm for reverse sampling $\mathcal{Y}$ as in Definition 11.

4.2 What is the weakest "useful" notion of a PCF?

The weakest flavour of PCF in the literature is that of *weak PCF*. Assume two parties wish to use a weak PCF for OT correlations[13] in order to generate correlated randomness to be used for secure

[13] A 1-out-of-2 bit-OT correlation can be defined as being sampled as a pair (of pairs) (m_0, m_1) and (σ, m_σ), where (m_0, m_1) are the OT sender's random messages in $\{0,1\}$, and σ is the random choice bit given to the receiver.

computation. They will need some way to agree on which OT correlations to use, *i.e.* on which points their PCF keys should be evaluated. If they were using a non-adaptive PCF, they could simply use some predetermined order, *e.g.* 1, 2, 3, *etc.* or $(\mathsf{sid}, 1)$, $(\mathsf{sid}, 2)$, $(\mathsf{sid}, 3)$, *etc.* for a session identifier sid. However, with a *weak* PCF the OT correlations will only be guaranteed to be "safe to use"[14] when indices are chosen *uniformly* at random. Thus, the parties need to agree beforehand on a random string or a CRS which grows with the size of the computation. This raises the following question:

> Is there an intermediary notion, stronger than a wPCF but weaker than a naPCF, which is directly useful for MPC applications without a CRS?

A natural idea is to replace a large random string by a *pseudorandom* string which can be generated by a pseudorandom function using a small seed which is small enough to become a part of each party's PCF key. We thus introduce the concept of a *pseudorandom-input* PCF (PI-PCF), which remains correct and secure even if the PCF inputs are chosen pseudorandomly, according to a *public* seed.

4.2.1 Defining a Pseudorandom-Input PCF (PI-PCF).

Again, we rely on the concept of an *admissible* sampler (Definition 2), which we previously introduced in the context of PI-PRFs.
We define a PI-PCF syntactically in the same way as a weak PCF (Definition 12), but demand the stronger properties of *pseudorandom $\mathcal{Y}$-correlated outputs* and *PCF security*, which we describe in Definitions 19 and 20 (differences with the corresponding notions for a weak PCF are highlighted). We refer to **??** for a complete and standalone definition.

Definition 19 (Pseudorandom $\mathcal{Y}$-correlated outputs of a PI-PCF). *For every non-uniform PPT adversary $\mathcal{A}$, it holds that for all polynomials N, for all admissible samplers $\mathsf{Sam}_{n(\lambda),N}$,*

$$|\Pr[\mathsf{Exp}^{\mathsf{PI\text{-}pr}}_{\mathcal{A},N,0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{PI\text{-}pr}}_{\mathcal{A},N,1}(\lambda) = 1]|$$

is negligible, where Figure 11 defines $\mathsf{Exp}^{\mathsf{PI\text{-}pr}}_{\mathcal{A},N,b}(\lambda)$ ($b \in \{0,1\}$).

Experiment Pseudorandom Correlated Outputs for Pseudorandom Inputs

```
$\mathsf{Exp}^{\mathsf{PI\text{-}pr}}_{\mathcal{A},N,0}(\lambda)$

$r \leftarrow\$ \{0,1\}^\lambda$
$(x^{(1)}, \ldots, x^{(N(\lambda))}) \leftarrow \mathsf{Sam}_{n(\lambda),N(\lambda)}(r)$
for $i = 1 \ldots N(\lambda)$ :
    $(y_0^{(i)}, y_1^{(i)}) \hookleftarrow \mathcal{Y}(1^\lambda)$
$b \leftarrow \mathcal{A}(1^\lambda, (x^{(i)}, y_0^{(i)}, y_1^{(i)})_{i \in [N(\lambda)]}, r)$
return $b$
```

```
$\mathsf{Exp}^{\mathsf{PI\text{-}pr}}_{\mathcal{A},N,1}(\lambda)$
$(k_0, k_1) \leftarrow\$ \mathsf{PI\text{-}PCF.Gen}(1^\lambda)$
$r \leftarrow\$ \{0,1\}^\lambda$
$(x^{(1)}, \ldots, x^{(N(\lambda))}) \leftarrow \mathsf{Sam}_{n(\lambda),N(\lambda)}(r)$
for $i = 1 \ldots N(\lambda)$ :
    for $\sigma \in \{0,1\}$ :
        $y_\sigma^{(i)} \leftarrow\$ \mathsf{PI\text{-}PCF.Eval}(\sigma, k_\sigma, x^{(i)})$
$b \leftarrow \mathcal{A}(1^\lambda, (x^{(i)}, y_0^{(i)}, y_1^{(i)})_{i \in [N(\lambda)]}, r)$
return $b$
```

Fig. 11: Pseudorandom $\mathcal{Y}$-correlated outputs of a PI-PCF. Differences with a wPCF (Figure 5) are highlighted.

Definition 20 (PI-PCF Security). *For every $\sigma \in \{0,1\}$ and every non-uniform PPT $\mathcal{A}$, it holds that for all polynomial N, for all admissible samplers $\mathsf{Sam}_{n(\lambda),N}$,*

$$|\Pr[\mathsf{Exp}^{\mathsf{PI\text{-}sec}}_{\mathcal{A},N,\sigma,0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{PI\text{-}sec}}_{\mathcal{A},N,\sigma,1}(\lambda) = 1]|$$

[14] More precisely, *correctness* (*i.e.* parties hold tuples of the form (m_0, m_1) and (σ, m_σ)) is tied to the wPCF having *(weakly) OT-correlated pseudorandom outputs*, while security (*i.e.* $m_{1-\sigma}$ is hidden from the receiver and σ is hidden from the sender) is tied to *wPCF security*.

is negligible, where $\mathsf{Exp}^{\mathsf{PI\text{-}sec}}_{\mathcal{A},N,\sigma,b}$ *($b \in \{0,1\}$) is defined as in Figure 12.*

```
***Experiment*** PI-PCF Security

$\mathsf{Exp}^{\mathsf{PI\text{-}sec}}_{\mathcal{A},N,\sigma,0}(\lambda)$
$(k_0, k_1) \leftarrow\$ \mathsf{PI\text{-}PCF.Gen}(1^\lambda)$
$r \leftarrow\$ \{0,1\}^\lambda$
$(x^{(1)}, \ldots, x^{(N(\lambda))}) \leftarrow \mathsf{Sam}_{n(\lambda),N(\lambda)}(r)$
for $i = 1 \ldots N(\lambda)$ :
    $y^{(i)}_{1-\sigma} \leftarrow\$ \mathsf{PI\text{-}PCF.Eval}(1-\sigma, k_{1-\sigma}, x^{(i)})$
$b \leftarrow \mathcal{A}(1^\lambda, \sigma, k_\sigma, (x^{(i)}, y^{(i)}_{1-\sigma})_{i \in [N(\lambda)]}, r)$
return $b$

$\mathsf{Exp}^{\mathsf{PI\text{-}sec}}_{\mathcal{A},N,\sigma,1}(\lambda)$
$(k_0, k_1) \leftarrow\$ \mathsf{PI\text{-}PCF.Gen}(1^\lambda)$
$r \leftarrow\$ \{0,1\}^\lambda$
$(x^{(1)}, \ldots, x^{(N(\lambda))}) \leftarrow \mathsf{Sam}_{n(\lambda),N(\lambda)}(r)$
for $i = 1 \ldots N(\lambda)$ :
    $y^{(i)}_{\sigma} \leftarrow\$ \mathsf{PI\text{-}PCF.Eval}(\sigma, k_{\sigma}, x^{(i)})$
    $y^{(i)}_{1-\sigma} \leftarrow\$ \mathsf{RSample}(1^\lambda, \sigma, y^{(i)}_{\sigma})$
$b \leftarrow \mathcal{A}(1^\lambda, \sigma, k_\sigma, (x^{(i)}, y^{(i)}_{1-\sigma})_{i \in [N(\lambda)]}, r)$
return $b$
```

Fig. 12: Security of a pseudorandom-input PCF. Here, RSample is the algorithm for reverse sampling $\mathcal{Y}$ as in Definition 11.

We note that the requirement of tolerating *any* admissible sampler is a strong one (but admittedly still weaker than a non-adaptive PCF). We discuss in Remark 24 a meaningful relaxation which is still strong enough to allow our transformation to go through.

4.2.2 A conditional argument towards minimality.

Let us now show that if there exists a weak PCF which is *not* a pseudorandom-input PCF, then it can be used to build an infinitely-often key-agreement scheme. This theorem is not trivial in the sense that a weak PCF can be seen a form of *silent (and incremental) OT extension*, and is not known to imply the existence of infinitely often key-agreement[15]. Moreover, as we discuss in Section 5.2, there are plausible candidates of weak PCFs from assumptions not known to imply infinitely often key-agreement.

Theorem 21 (wPCF not PI-PCF implies io-KA). *Let* wPCF *be a weak PCF (for some correlation). If* wPCF *is* not *a pseudorandom-input PCF (for that same correlation), then there exists an infinitely often two-party key-agreement protocol.*

Proof sketch. If wPCF is a weak PCF for some correlation $\mathcal{Y}$, but *not* a pseudorandom-input PCF for $\mathcal{Y}$, then this means that at least one out of (1) *pseudorandom-input pseudorandom $\mathcal{Y}$-correlated outputs* or (2) the *pseudorandom-input PCF security* is violated. The proof proceeds by case distinction and shows that in either case, there is a key-agreement protocol which $\frac{1}{2} + \epsilon$, for a non-negligible function ϵ. The argument is analogous to Pietrak-Sjödin [PS08]. For completeness, we include the proofs of both statements in Appendix B.2.

4.2.3 Defining a fully non-adaptive PCF (fnaPCF).

The notion of PI-PCF we just introduced is analogous to the notion of PI-PRF we introduced in Definition 3. In Section 3.2, we showed a modular strengthening from PI-PRF to sPRF, via a naPRF; it is therefore a natural question to ask whether the same transformation can be used to first turn a PI-PCF into a naPCF, and then a naPCF into a sPCF. This would be interesting because the first transform, which does not even require the application of an ELF, has the potential of being very lightweight. Unfortunately, for technical reasons, the intermediary notion we obtain is not a non-adaptive PCF, but what we coin as a *fully non-adaptive PCF*. The difference lies in the inputs of a non-adaptive PCF must be chosen before seeing any of the evaluations of the honest party's PCF key, but can be chosen *after* seeing the corrupt party's PCF key.

[15] In fact, (interactive) OT extension is known to be in Minicrypt [IKNP03]. The minimal assumptions for silent OT extension are unknown.

We now introduce the notion of a *fully non-adaptive PCF (fnaPCF)*, which differs from a non-adaptive PCF in that in the PCF security game, the adversary must produce the evaluation points before even seeing the corrupt party's PCF key. A fnaPCF is syntactically defined a non-adaptive PCF (**??**) and satisfies the same notion of *non-adaptively pseudorandom $\mathcal{Y}$-correlated outputs* (Definition 15) as a non-adaptive PCF, but satisfies a stronger security property which we define in Definition 22 (differences with the security of a naPCF are highlighted).

Definition 22 (Fully Non-Adaptive PCF Security). *For every $\sigma \in \{0,1\}$ and every non-uniform adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ of size $B(\lambda)$, it holds that for all sufficiently large λ,*

$$|\Pr[\mathsf{Exp}^{\mathsf{fna\text{-}sec}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),N,\sigma,0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{fna\text{-}sec}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),N,\sigma,1}(\lambda) = 1]| \leq \epsilon(\lambda)$$

where $\mathsf{Exp}^{\mathsf{na\text{-}sec}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),N,\sigma,b}$ ($b \in \{0,1\}$) is defined as in Figure 13.

Experiment Fully Non-Adaptive PCF Security

```
$\mathsf{Exp}^{\mathsf{fna\text{-}sec}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),N,\sigma,0}(\lambda)$
$(k_0, k_1) \leftarrow\$ \mathsf{fnaPCF.Gen}(1^\lambda)$
$((x^{(i)})_{i\in[N(\lambda)]}, \mathsf{state}) \leftarrow\$ \mathcal{A}_0(1^\lambda, \sigma)$
for $i = 1$ to $N(\lambda)$ :

    $y^{(i)}_{1-\sigma} \leftarrow \mathsf{fnaPCF.Eval}(1-\sigma, k_{1-\sigma}, x^{(i)})$
$b \leftarrow\$ \mathcal{A}_1(1^\lambda, \sigma, \mathsf{state}, k_\sigma, (x^{(i)}, y^{(i)}_{1-\sigma})_{i\in[N(\lambda)]})$
return $b$
```

```
$\mathsf{Exp}^{\mathsf{fna\text{-}sec}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),N,\sigma,1}(\lambda)$
$(k_0, k_1) \leftarrow\$ \mathsf{fnaPCF.Gen}(1^\lambda)$
$((x^{(i)})_{i\in[N(\lambda)]}, \mathsf{state}) \leftarrow\$ \mathcal{A}_0(1^\lambda, \sigma)$
for $i = 1$ to $N(\lambda)$ :
    $y^{(i)}_{\sigma} \leftarrow \mathsf{fnaPCF.Eval}(\sigma, k_\sigma, x^{(i)})$
    $y^{(i)}_{1-\sigma} \leftarrow\$ \mathsf{RSample}(1^\lambda, \sigma, y^{(i)}_\sigma)$
$b \leftarrow\$ \mathcal{A}_1(1^\lambda, \sigma, \mathsf{state}, k_\sigma, (x^{(i)}, y^{(i)}_{1-\sigma})_{i\in[N(\lambda)]})$
return $b$
```

Fig. 13: Security of a fully non-adaptive PCF. Here, $\mathsf{RSample}$ is the algorithm for reverse sampling $\mathcal{Y}$ as in Definition 11. Differences with Figure 13 are highlighted.

4.3 Boosting security from PI-PCF to fnaPCF.

Having defined the notions of pseudorandom-input and fully non-adaptive PCFs, we are ready to introduce our transform.

fnaPCF Fully Non-Adaptive PCF from Pseudorandom-Input PCF + naPRF

`Requires:`

- $\mathcal{Y}$ is a reverse-sampleable correlation with input length function $n(\lambda)$.
- $\mathsf{PI\text{-}PCF} = (\mathsf{PI\text{-}PCF.Gen}, \mathsf{PI\text{-}PCF.Eval})$ is a pseudorandom-input weak PCF for $\mathcal{Y}$.
- naPRF is a non-adaptive PRF with key space $\{0,1\}^\lambda$, input space $\{0,1\}^{n(\lambda)}$, and output space $\{0,1\}^{n(\lambda)}$.

```
$\mathsf{fnaPCF.Gen}(1^\lambda)$:
1. $k_{\mathsf{naPRF}} \leftarrow\$ \{0,1\}^\lambda$
2. $(k_0^{\mathsf{PI\text{-}PCF}}, k_1^{\mathsf{PI\text{-}PCF}}) \leftarrow\$ \mathsf{PI\text{-}PCF.Gen}(1^\lambda)$
3. For $\sigma \in \{0,1\}$, set $k_\sigma \leftarrow (k_{\mathsf{naPRF}}, k_\sigma^{\mathsf{PI\text{-}PCF}})$
4. Output $(k_0, k_1)$
```

```
$\mathsf{fnaPCF.Eval}(\sigma, k_\sigma, x)$:
1. Parse $k_\sigma$ as $k_\sigma = (k_{\mathsf{naPRF}}, k_\sigma^{\mathsf{PI\text{-}PCF}})$
2. $x' \leftarrow \mathsf{naPRF}(k_{\mathsf{naPRF}}, x)$
3. Set $y_\sigma \leftarrow \mathsf{PI\text{-}PCF.Eval}(\sigma, k_\sigma^{\mathsf{PI\text{-}PCF}}, x')$
4. Output $y_\sigma$
```

Fig. 14: Applying a non-adaptive PRF to a pseudorandom-input PCF's input yields a fully non-adaptive PCF.

Lemma 23 (PI-PCF ∘ naPRF is a fnaPCF). *Applying a non-adaptive PRF to the input of a pseudorandom-input weak PCF for some correlation $\mathcal{Y}$ yields a non-adaptive PCF for the same correlation $\mathcal{Y}$. More formally, the construction of Figure 14 is a non-adaptive PCF.*

Remark 24 (A PI-PCF "for all admissible samplers" is not required). By careful inspection of the proof of Lemma 23 (and specifically in the hop from hybrid Hyb_1 to Hyb_2 in Figure 15, and in the same hop in Figure 16), one may observe that all is required of the PI-PCF is that it tolerates admissible samplers of the form "$\mathsf{Sam}_{n(\lambda),N(\lambda)}$: $(x^{(i)})_{i\in[N(\lambda)]} \leftarrow\!\$\ \mathcal{A}(1^\lambda)$; $k_{\mathsf{naPRF}} \leftarrow\!\$\ \{0,1\}^\lambda$; For $i \in [N(\lambda)]$, $(x')^{(i)} \leftarrow \mathsf{naPRF}(k_{\mathsf{naPRF}}, x^{(i)})$; Return $((x')^{(i)})_{i\in[N(\lambda)]}$". This relaxation on the notion of a PI-PCF is the key to plausibly instantiating it under variants of LPN, as discussed in Section 5.2.

Proof of Lemma 23. Let $\mathcal{Y}$ be a reverse-sampleable correlation with input length function $n(\lambda)$, let $\mathsf{PI\text{-}PCF} = (\mathsf{PI\text{-}PCF.Gen}, \mathsf{PI\text{-}PCF.Eval})$ be a pseudorandom-input weak PCF for $\mathcal{Y}$, let naPRF be a non-adaptive PRF with key space $\{0,1\}^\lambda$, input space $\{0,1\}^{n(\lambda)}$, and output space $\{0,1\}^{n(\lambda)}$. Let $\mathsf{fnaPCF} = (\mathsf{fnaPCF.Gen}, \mathsf{fnaPCF.Eval})$ be defined as in Figure 14.
In order to show that it is a fully non-adaptive PCF, we need to show it has *non-adaptively pseudorandom $\mathcal{Y}$-correlated outputs* and that it satisfies *fully non-adaptive PCF security.* Both reductions follow along the same lines as the proof of Lemma 5, showing an analogous transformation from pseudorandom-input PRF to non-adaptive PRF.

1. *Non-adaptively pseudorandom $\mathcal{Y}$-correlated outputs.* Let N be a polynomial function, and let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ be a non-uniform adversary of size $B(\lambda)$ making at most $N(\lambda)$ non-adaptive queries. Without loss of generality, we assume that the state output by $\mathcal{A}_0$ are the random coins it used (recall that $\mathcal{A}_0$ outputs $N(\lambda)$ naPRF inputs as well as its state, to be passed to the distinguisher $\mathcal{A}_1$); we denote ℓ_0 the length of this state/randomness.
   Consider the sequence of hybrids $\mathsf{Hyb}_0, \mathsf{Hyb}_1, \mathsf{Hyb}_2, \mathsf{Hyb}_3$ as defined in Figure 15.
   We now show these hybrids to be indistinguishable.
   - $H_0 \equiv H_1$: This follows from the observation that H_0 and H_1 are code-equivalent; we simply inlined the definitions of $\mathsf{fnaPCF.Gen}$ and $\mathsf{fnaPCF.Eval}$, then introduced $\mathsf{Sam}_{n(\lambda),N(\lambda)}$.
   - $H_1 \overset{c}{\approx} H_2$: By security of the non-adaptive PRF naPRF, the sampler $\mathsf{Sam}_{n(\lambda),N(\lambda)}$ is admissible (Definition 2). Since the outputs of $\mathsf{PI\text{-}PCF}$ are *PI-pseudorandom $\mathcal{Y}$-correlated* and $\mathsf{Sam}_{n(\lambda),N(\lambda)}$ is admissible, $H_1 \overset{c}{\approx} H_2$.
   - $H_2 \equiv H_3$: H_2 and H_3 are code-equivalent (as the highlighted lines define random variables never subsequently used).
2. *Fully Non-adaptive PCF security.* Let $\sigma \in \{0,1\}$. Let N be a polynomial function, and let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ be a non-uniform adversary of size $B(\lambda)$ making at most $N(\lambda)$ non-adaptive queries. Without loss of generality, we assume that the state output by $\mathcal{A}_0$ are the random coins it used (recall that $\mathcal{A}_0$ outputs $N(\lambda)$ naPRF inputs as well as its state, to be passed to the distinguisher $\mathcal{A}_1$); we denote ℓ_0 the length of this state/randomness.
   Consider the sequence of hybrids $\mathsf{Hyb}_0, \mathsf{Hyb}_1, \mathsf{Hyb}_2, \mathsf{Hyb}_3$ as defined in Figure 16.
   We now show these hybrids to be indistinguishable.
   - $H_0 \equiv H_1$: This follows from the observation that H_0 and H_1 are code-equivalent.
   - $H_1 \overset{c}{\approx} H_2$: By security of the non-adaptive PRF naPRF, the sampler $\mathsf{Sam}_{n(\lambda),N(\lambda)}$ is admissible (Definition 2). By applying *PI-PCF security* of $\mathsf{PI\text{-}PCF}$, with the admissible sampler $\mathsf{Sam}_{n(\lambda),N(\lambda)}$, we immediately get that $H_1 \overset{c}{\approx} H_2$.
   - $H_2 \equiv H_3$: This follows from the observation that H_2 and H_3 are code-equivalent (Completely analogously to how H_1 and H_1, as these games use the same primitives of $\mathsf{Sam}_{n(\lambda),N(\lambda)}$, $\mathsf{fnaPCF.Gen}$, and $\mathsf{fnaPCF.Eval}$).

□

4.4 Boosting security from fnaPCF to sPCF.

We now show a transform from a fully non-adaptive to a strong PCF.

```
$\mathsf{Hyb}_0 = \mathsf{Exp}^{\text{na-pr}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),N,1}(\lambda)$

$((x^{(i)})_{i\in[N(\lambda)]}, \mathsf{state}) \leftarrow\!\$\ \mathcal{A}_0(1^\lambda)$
$(k_0, k_1) \leftarrow\!\$\ \mathsf{fnaPCF.Gen}(1^\lambda)$
for $i = 1$ to $N(\lambda)$ :
   for $\sigma \in \{0,1\}$ :
      $y_\sigma^{(i)} \leftarrow \mathsf{fnaPCF.Eval}(\sigma, k_\sigma, x^{(i)})$
$b \leftarrow\!\$\ \mathcal{A}_1(\mathsf{state}, (y_0^{(i)}, y_1^{(i)})_{i\in[N(\lambda)]})$
return $b$

$\mathsf{fnaPCF.Gen}(1^\lambda)$

$k_{\mathsf{naPRF}} \leftarrow\!\$\ \{0,1\}^\lambda$
$(k_0^{\mathsf{PI\text{-}PCF}}, k_1^{\mathsf{PI\text{-}PCF}}) \leftarrow\!\$\ \mathsf{PI\text{-}PCF.Gen}(1^\lambda)$
for $\sigma \in \{0,1\}$ :
   $k_\sigma \leftarrow (k_{\mathsf{naPRF}}, k_\sigma^{\mathsf{PI\text{-}PCF}})$
return $(k_0, k_1)$

$\mathsf{fnaPCF.Eval}(\sigma, k_\sigma, x)$

Parse $k_\sigma = (k_{\mathsf{naPRF}}, k_\sigma^{\mathsf{PI\text{-}PCF}})$
$x' \leftarrow \mathsf{naPRF}(k_{\mathsf{naPRF}}, x)$
$y_\sigma \leftarrow \mathsf{PI\text{-}PCF.Eval}(\sigma, k_\sigma^{\mathsf{PI\text{-}PCF}}, x')$
return $y_\sigma$
```

```
$\mathsf{Hyb}_1(\lambda)$

$r \leftarrow\!\$\ \{0,1\}^{\lambda+\ell_0}$
$((x')^{(i)})_{i\in[N(\lambda)]} \leftarrow \mathsf{Sam}_{n(\lambda),N(\lambda)}(r)$
$(k_0^{\mathsf{PI\text{-}PCF}}, k_1^{\mathsf{PI\text{-}PCF}}) \leftarrow\!\$\ \mathsf{PI\text{-}PCF.Gen}(1^\lambda)$
for $i = 1$ to $N(\lambda)$ :
   for $\sigma \in \{0,1\}$ :
      $y_\sigma^{(i)} \leftarrow \mathsf{PI\text{-}PCF.Eval}(\sigma, k_\sigma^{\mathsf{PI\text{-}PCF}}, (x')^{(i)})$
$b \leftarrow\!\$\ \mathcal{A}_1(r_0, (y_0^{(i)}, y_1^{(i)})_{i\in[N(\lambda)]})$
return $b$

$\mathsf{Sam}_{n(\lambda),N(\lambda)}(r)$ :

$k_{\mathsf{naPRF}} \leftarrow r[0, \lambda-1]$
$r_0 \leftarrow r[\lambda, \lambda+\ell_0-1]$
$(x^{(1)}, \ldots, x^{(N(\lambda))}) \leftarrow\!\$\ \mathcal{A}_0(1^\lambda; r_0)$
for $i = 1$ to $N(\lambda)$ :
   $(x')^{(i)} \leftarrow \mathsf{naPRF}(k_{\mathsf{naPRF}}, x^{(i)})$
return $((x')^{(1)}, \ldots, (x')^{(N(\lambda))})$
```

```
$\mathsf{Hyb}_2(\lambda)$

$r \leftarrow\!\$\ \{0,1\}^{\lambda+\ell_0}$
$((x')^{(i)})_{i\in[N(\lambda)]} \leftarrow \mathsf{Sam}_{n(\lambda),N(\lambda)}(r)$
$(k_0^{\mathsf{PI\text{-}PCF}}, k_1^{\mathsf{PI\text{-}PCF}}) \leftarrow\!\$\ \mathsf{PI\text{-}PCF.Gen}(1^\lambda)$
for $i = 1$ to $N(\lambda)$ :

      $(y_0^{(i)}, y_1^{(i)}) \leftarrow\!\$\ \mathcal{Y}$
$b \leftarrow\!\$\ \mathcal{A}_1(r_0, (y_0^{(i)}, y_1^{(i)})_{i\in[N(\lambda)]})$
return $b$

$\mathsf{Sam}_{n(\lambda),N(\lambda)}(r)$ :

$k_{\mathsf{naPRF}} \leftarrow r[0, \lambda-1]$
$r_0 \leftarrow r[\lambda, \lambda+\ell_0-1]$
$(x^{(1)}, \ldots, x^{(N(\lambda))}) \leftarrow\!\$\ \mathcal{A}_0(1^\lambda; r_0)$
for $i = 1$ to $N(\lambda)$ :
   $(x')^{(i)} \leftarrow \mathsf{naPRF}(k_{\mathsf{naPRF}}, x^{(i)})$
return $((x')^{(1)}, \ldots, (x')^{(N(\lambda))})$
```

```
$\mathsf{Hyb}_2(\lambda)$ (repeated)

$r \leftarrow\!\$\ \{0,1\}^{\lambda+\ell_0}$
$r_0 \leftarrow r[\lambda, \lambda+\ell_0-1]$
$((x')^{(i)})_{i\in[N(\lambda)]} \leftarrow \mathsf{Sam}_{n(\lambda),N(\lambda)}(r)$
$(k_0^{\mathsf{PI\text{-}PCF}}, k_1^{\mathsf{PI\text{-}PCF}}) \leftarrow\!\$\ \mathsf{PI\text{-}PCF.Gen}(1^\lambda)$
for $i = 1$ to $N(\lambda)$ :

   $(y_0^{(i)}, y_1^{(i)}) \leftarrow\!\$\ \mathcal{Y}$
$b \leftarrow\!\$\ \mathcal{A}_1(r_0, (y_0^{(i)}, y_1^{(i)})_{i\in[N(\lambda)]})$
return $b$

$\mathsf{Sam}_{n(\lambda),N(\lambda)}(r)$ :

$k_{\mathsf{naPRF}} \leftarrow r[0, \lambda-1]$
$r_0 \leftarrow r[\lambda, \lambda+\ell_0-1]$
$(x^{(1)}, \ldots, x^{(N(\lambda))}) \leftarrow\!\$\ \mathcal{A}_0(1^\lambda; r_0)$
for $i = 1$ to $N(\lambda)$ :
   $(x')^{(i)} \leftarrow \mathsf{naPRF}(k_{\mathsf{naPRF}}, x^{(i)})$
return $((x')^{(1)}, \ldots, (x')^{(N(\lambda))})$
```

```
$\mathsf{Hyb}_3 = \mathsf{Exp}^{\text{na-pr}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),N,0}(\lambda)$

$((x^{(i)})_{i\in[N(\lambda)]}, \mathsf{state}) \leftarrow\!\$\ \mathcal{A}_0(1^\lambda)$



for $i = 1$ to $N(\lambda)$ :

   $(y_0^{(i)}, y_1^{(i)}) \leftarrow\!\$\ \mathcal{Y}$
$b \leftarrow\!\$\ \mathcal{A}_1(\mathsf{state}, (y_0^{(i)}, y_1^{(i)})_{i\in[N(\lambda)]})$
return $b$

$\mathsf{fnaPCF.Gen}(1^\lambda)$

$k_{\mathsf{naPRF}} \leftarrow\!\$\ \{0,1\}^\lambda$
$(k_0^{\mathsf{PI\text{-}PCF}}, k_1^{\mathsf{PI\text{-}PCF}}) \leftarrow\!\$\ \mathsf{PI\text{-}PCF.Gen}(1^\lambda)$
for $\sigma \in \{0,1\}$ :
   $k_\sigma \leftarrow (k_{\mathsf{naPRF}}, k_\sigma^{\mathsf{PI\text{-}PCF}})$
return $(k_0, k_1)$

$\mathsf{fnaPCF.Eval}(\sigma, k_\sigma, x)$

Parse $k_\sigma = (k_{\mathsf{naPRF}}, k_\sigma^{\mathsf{PI\text{-}PCF}})$
$x' \leftarrow \mathsf{naPRF}(k_{\mathsf{naPRF}}, x)$
$y_\sigma \leftarrow \mathsf{PI\text{-}PCF.Eval}(\sigma, k_\sigma^{\mathsf{PI\text{-}PCF}}, x')$
return $y_\sigma$
```

Fig. 15: Sequence of hybrids for proving *non-adaptively pseudorandom $\mathcal{Y}$-correlated outputs* in the proof of Lemma 23.

sPCF Strong PCF from Fully Non-Adaptive PCF + ELF

```
Requires:
```

```
$\mathsf{Hyb}_0 = \mathsf{Exp}^{\mathsf{fna\text{-}sec}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),N,\sigma,0}(\lambda)$
$(k_0, k_1) \leftarrow\$ \mathsf{fnaPCF.Gen}(1^\lambda)$
$((x^{(i)})_{i\in[N(\lambda)]}, \mathsf{state}) \leftarrow\$ \mathcal{A}_0(1^\lambda, \sigma)$
for $i = 1$ to $N(\lambda)$ :
    $y^{(i)}_{1-\sigma} \leftarrow \mathsf{fnaPCF.Eval}(1-\sigma, k_{1-\sigma}, x^{(i)})$
$b \leftarrow\$ \mathcal{A}_1(1^\lambda, \sigma, \mathsf{state}, k_\sigma, (x^{(i)}, y^{(i)}_{1-\sigma})_{i\in[N(\lambda)]})$
return $b$

$\mathsf{fnaPCF.Gen}(1^\lambda)$
$k_{\mathsf{naPRF}} \leftarrow\$ \{0,1\}^\lambda$
$(k_0^{\mathsf{PI\text{-}PCF}}, k_1^{\mathsf{PI\text{-}PCF}}) \leftarrow\$ \mathsf{PI\text{-}PCF.Gen}(1^\lambda)$
for $\sigma \in \{0,1\}$ :
    $k_\sigma \leftarrow (k_{\mathsf{naPRF}}, k_\sigma^{\mathsf{PI\text{-}PCF}})$
return $(k_0, k_1)$

$\mathsf{fnaPCF.Eval}(\sigma, k_\sigma, x)$
Parse $k_\sigma = (k_{\mathsf{naPRF}}, k_\sigma^{\mathsf{PI\text{-}PCF}})$
$x' \leftarrow \mathsf{naPRF}(k_{\mathsf{naPRF}}, x)$
$y_\sigma \leftarrow \mathsf{PI\text{-}PCF.Eval}(\sigma, k_\sigma^{\mathsf{PI\text{-}PCF}}, x')$
return $y_\sigma$
```

```
$\mathsf{Hyb}_1(\lambda)$
$(k_0^{\mathsf{PI\text{-}PCF}}, k_1^{\mathsf{PI\text{-}PCF}}) \leftarrow\$ \mathsf{PI\text{-}PCF.Gen}(1^\lambda)$
$r \leftarrow\$ \{0,1\}^{\lambda+\ell_0}$
$r_0 \leftarrow r[\lambda, \lambda+\ell_0-1]$; $r_{\mathsf{naPRF}} \leftarrow r[0, \lambda-1]$
$(x^{(i)})_{i\in[N(\lambda)]} \leftarrow \mathsf{Sam}_{n(\lambda),N(\lambda)}(r)$
for $i = 1$ to $N(\lambda)$ :
    $y^{(i)}_{1-\sigma} \leftarrow \mathsf{PI\text{-}PCF.Eval}(\sigma, k_{1-\sigma}^{\mathsf{PI\text{-}PCF}}, x^{(i)})$
$b \leftarrow\$ \mathcal{A}_1(1^\lambda, \sigma, r_0, r_1, k_\sigma^{\mathsf{PI\text{-}PCF}}, (x^{(i)}, y^{(i)}_{1-\sigma})_{i\in[N(\lambda)]})$
return $b$

$\mathsf{Sam}_{n(\lambda),N(\lambda)}(r)$ :
$k_{\mathsf{naPRF}} \leftarrow r[0, \lambda-1]$
$r_0 \leftarrow r[\lambda, \lambda+\ell_0-1]$
$(x^{(1)}, \dots, x^{(N(\lambda))}) \leftarrow\$ \mathcal{A}_0(1^\lambda; r_0)$
for $i = 1$ to $N(\lambda)$ :
    $(x')^{(i)} \leftarrow \mathsf{naPRF}(k_{\mathsf{naPRF}}, x^{(i)})$
return $((x')^{(1)}, \dots, (x')^{(N(\lambda))})$
```

```
$\mathsf{Hyb}_2(\lambda)$
$(k_0^{\mathsf{PI\text{-}PCF}}, k_1^{\mathsf{PI\text{-}PCF}}) \leftarrow\$ \mathsf{PI\text{-}PCF.Gen}(1^\lambda)$
$r \leftarrow\$ \{0,1\}^{\lambda+\ell_0}$
$r_0 \leftarrow r[\lambda, \lambda+\ell_0-1]$; $r_{\mathsf{naPRF}} \leftarrow r[0, \lambda-1]$
$(x^{(i)})_{i\in[N(\lambda)]} \leftarrow \mathsf{Sam}_{n(\lambda),N(\lambda)}(r)$
for $i = 1$ to $N(\lambda)$ :
    $y^{(i)}_{\sigma} \leftarrow \mathsf{PI\text{-}PCF.Eval}(\sigma, k_{\sigma}^{\mathsf{PI\text{-}PCF}}, (x')^{(i)})$
    $y^{(i)}_{1-\sigma} \leftarrow\$ \mathsf{RSample}(1^\lambda, \sigma, y^{(i)}_\sigma)$
$b \leftarrow\$ \mathcal{A}_1(1^\lambda, \sigma, r_0, r_1, k_\sigma^{\mathsf{PI\text{-}PCF}}, (x^{(i)}, y^{(i)}_{1-\sigma})_{i\in[N(\lambda)]})$
return $b$

$\mathsf{Sam}_{n(\lambda),N(\lambda)}(r)$ :
$k_{\mathsf{naPRF}} \leftarrow r[0, \lambda-1]$
$r_0 \leftarrow r[\lambda, \lambda+\ell_0-1]$
$(x^{(1)}, \dots, x^{(N(\lambda))}) \leftarrow\$ \mathcal{A}_0(1^\lambda; r_0)$
for $i = 1$ to $N(\lambda)$ :
    $(x')^{(i)} \leftarrow \mathsf{naPRF}(k_{\mathsf{naPRF}}, x^{(i)})$
return $((x')^{(1)}, \dots, (x')^{(N(\lambda))})$
```

```
$\mathsf{Hyb}_2(\lambda)$
$(k_0^{\mathsf{PI\text{-}PCF}}, k_1^{\mathsf{PI\text{-}PCF}}) \leftarrow\$ \mathsf{PI\text{-}PCF.Gen}(1^\lambda)$
$r \leftarrow\$ \{0,1\}^{\lambda+\ell_0}$
$r_0 \leftarrow r[\lambda, \lambda+\ell_0-1]$; $r_{\mathsf{naPRF}} \leftarrow r[0, \lambda-1]$
$((x')^{(i)})_{i\in[N(\lambda)]} \leftarrow \mathsf{Sam}_{n(\lambda),N(\lambda)}(r)$
for $i = 1$ to $N(\lambda)$ :
    $y^{(i)}_{\sigma} \leftarrow \mathsf{PI\text{-}PCF.Eval}(\sigma, k_{\sigma}^{\mathsf{PI\text{-}PCF}}, (x')^{(i)})$
    $y^{(i)}_{1-\sigma} \leftarrow\$ \mathsf{RSample}(1^\lambda, \sigma, y^{(i)}_\sigma)$
$b \leftarrow\$ \mathcal{A}_1(1^\lambda, \sigma, r_0, r_1, k_\sigma^{\mathsf{PI\text{-}PCF}}, (x^{(i)}, y^{(i)}_{1-\sigma})_{i\in[N(\lambda)]})$
return $b$

$\mathsf{Sam}_{n(\lambda),N(\lambda)}(r)$ :
$k_{\mathsf{naPRF}} \leftarrow r[0, \lambda-1]$
$r_0 \leftarrow r[\lambda, \lambda+\ell_0-1]$
$(x^{(1)}, \dots, x^{(N(\lambda))}) \leftarrow\$ \mathcal{A}_0(1^\lambda; r_0)$
for $i = 1$ to $N(\lambda)$ :
    $(x')^{(i)} \leftarrow \mathsf{naPRF}(k_{\mathsf{naPRF}}, x^{(i)})$
return $((x')^{(1)}, \dots, (x')^{(N(\lambda))})$
```

```
$\mathsf{Hyb}_3 = \mathsf{Exp}^{\mathsf{fna\text{-}sec}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),N,\sigma,1}(\lambda)$
$(k_0, k_1) \leftarrow\$ \mathsf{fnaPCF.Gen}(1^\lambda)$
$((x^{(i)})_{i\in[N(\lambda)]}, \mathsf{state}) \leftarrow\$ \mathcal{A}_0(1^\lambda, \sigma)$
for $i = 1$ to $N(\lambda)$ :
    $y^{(i)}_{\sigma} \leftarrow \mathsf{fnaPCF.Eval}(\sigma, k_{\sigma}, x^{(i)})$
    $y^{(i)}_{1-\sigma} \leftarrow\$ \mathsf{RSample}(1^\lambda, \sigma, y^{(i)}_\sigma)$
$b \leftarrow\$ \mathcal{A}_1(1^\lambda, \sigma, \mathsf{state}, k_\sigma, (x^{(i)}, y^{(i)}_{1-\sigma})_{i\in[N(\lambda)]})$
return $b$

$\mathsf{fnaPCF.Gen}(1^\lambda)$
$k_{\mathsf{naPRF}} \leftarrow\$ \{0,1\}^\lambda$
$(k_0^{\mathsf{PI\text{-}PCF}}, k_1^{\mathsf{PI\text{-}PCF}}) \leftarrow\$ \mathsf{PI\text{-}PCF.Gen}(1^\lambda)$
for $\sigma \in \{0,1\}$ :
    $k_\sigma \leftarrow (k_{\mathsf{naPRF}}, k_\sigma^{\mathsf{PI\text{-}PCF}})$
return $(k_0, k_1)$

$\mathsf{fnaPCF.Eval}(\sigma, k_\sigma, x)$
Parse $k_\sigma = (k_{\mathsf{naPRF}}, k_\sigma^{\mathsf{PI\text{-}PCF}})$
$x' \leftarrow \mathsf{naPRF}(k_{\mathsf{naPRF}}, x)$
$y_\sigma \leftarrow \mathsf{PI\text{-}PCF.Eval}(\sigma, k_\sigma^{\mathsf{PI\text{-}PCF}}, x')$
return $y_\sigma$
```

Fig. 16: Sequence of hybrids for proving *fnaPCF security* in the proof of Lemma 23.

- $\mathcal{Y}$ is a reverse-sampleable correlation with input length function $n(\lambda)$ (we will be conflating the sets $[2^{n(\lambda)}]$ and $\{0,1\}^{n(\lambda)}$ via their natural bijection).
- $\mathsf{fnaPCF} = (\mathsf{fnaPCF.Gen}, \mathsf{fnaPCF.Eval})$ is a non-adaptive PCF for $\mathcal{Y}$.
- $\mathsf{ELF.Gen}$ is an extremely lossy function.

```
sPCF.Gen(1^λ):
  1. $f \leftarrow\!\$\ \mathsf{ELF.Gen}(2^{n(\lambda)}, 2^{n(\lambda)})$
  2. $(k_0^{\mathsf{fnaPCF}}, k_1^{\mathsf{fnaPCF}}) \leftarrow\!\$\ \mathsf{fnaPCF.Gen}(1^\lambda)$
  3. For $\sigma \in \{0,1\}$, set $k_\sigma \leftarrow (f, k_\sigma^{\mathsf{fnaPCF}})$
  4. Output $(k_0, k_1)$

sPCF.Eval(σ, k_σ, x):
  1. Parse $k_\sigma$ as $k_\sigma = (f, k_\sigma^{\mathsf{fnaPCF}})$
  2. $x' \leftarrow f(x)$
  3. Set $y_\sigma \leftarrow \mathsf{fnaPCF.Eval}(\sigma, k_\sigma^{\mathsf{fnaPCF}}, x')$
  4. Output $y_\sigma$
```

Fig. 17: Applying an ELF to a fully non-adaptive PCF's input yields a strong PCF.

Lemma 25 (fnaPCF ∘ ELF is a sPCF). *Applying an ELF to the input of a fully non-adaptive PCF for some correlation $\mathcal{Y}$ yields a strong PCF for the same correlation $\mathcal{Y}$. More formally, the construction of Figure 17 is a strong PCF.*

Proof. Let $\mathcal{Y}$ be a reverse-sampleable correlation with input length function $n(\lambda)$, let $\mathsf{fnaPCF} = (\mathsf{fnaPCF.Gen}, \mathsf{fnaPCF.Eval})$ be a pseudorandom-input weak PCF for $\mathcal{Y}$, let $\mathsf{ELF.Gen}$ be an extremely lossy function. Let $\mathsf{sPCF} = (\mathsf{sPCF.Gen}, \mathsf{sPCF.Eval})$ be defined as in Figure 17.
In order to show that it is a strong PCF, we need to show it has *strongly pseudorandom $\mathcal{Y}$-correlated outputs* and that it satisfies *strong PCF security*. Both reductions follow along the same lines as the proof of Lemma 6, showing an analogous transformation from non-adaptive PRF to strong PRF.

1. *Strongly pseudorandom $\mathcal{Y}$-correlated outputs.* Let $\mathcal{A}$ be an non-uniform adversary of size $B(\lambda)$ asking at most $N(\lambda)$ queries to the oracle $\mathcal{O}_b(\cdot)$ (as defined in Figure 9)
   Consider the sequence of hybrids $(\mathsf{Hyb}_i)_{i\in[0,9]}$, as defined in Figures 18 to 20.

$$\mathsf{Hyb}_0(1^\lambda) \xrightarrow{\text{inline } \mathsf{sPCF}} \mathsf{Hyb}_1(1^\lambda) \xrightarrow{\text{ELF indist.}} \mathsf{Hyb}_2(1^\lambda) \xrightarrow{\text{ELF enum. Im}}$$

```
Hyb_0(1^λ)
  $(k_0, k_1) \leftarrow\!\$\ \mathsf{sPCF.Gen}(1^\lambda)$
  $b \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}_1(\cdot)}(1^\lambda)$
  return $b$

O_1(x)
  for $\sigma \in \{0,1\}$ :
    $y_\sigma \leftarrow \mathsf{sPCF.Eval}(\sigma, k_\sigma, x)$
  return $(y_0, y_1)$
```

```
Hyb_1(1^λ)
  $f \leftarrow\!\$\ \mathsf{ELF.Gen}(2^{n(\lambda)}, 2^{n(\lambda)})$
  $(k_0^{\mathsf{fnaPCF}}, k_1^{\mathsf{fnaPCF}}) \leftarrow\!\$\ \mathsf{fnaPCF.Gen}(1^\lambda)$
  $b \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}_1(\cdot)}(1^\lambda)$
  return $b$

O_1(x)
  for $\sigma \in \{0,1\}$ :
    $y_\sigma \leftarrow \mathsf{fnaPCF.Eval}(\sigma, k_\sigma^{\mathsf{fnaPCF}}, f(x))$
  return $(y_0, y_1)$
```

```
Hyb_2(1^λ)
  $f \leftarrow\!\$\ \mathsf{ELF.Gen}(2^{n(\lambda)}, N(\lambda))$
  $(k_0^{\mathsf{fnaPCF}}, k_1^{\mathsf{fnaPCF}}) \leftarrow\!\$\ \mathsf{fnaPCF.Gen}(1^\lambda)$
  $b \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}_1(\cdot)}(1^\lambda)$
  return $b$

O_1(x)
  for $\sigma \in \{0,1\}$ :
    $y_\sigma \leftarrow \mathsf{fnaPCF.Eval}(\sigma, k_\sigma^{\mathsf{fnaPCF}}, f(x))$
  return $(y_0, y_1)$
```

Fig. 18: Sequence of hybrids for proving *strongly pseudorandom $\mathcal{Y}$-correlated outputs* in the proof of lemma 25 (Part 1/3). $\mathsf{Hyb}_0 = \mathsf{Exp}^{\mathsf{s\text{-}pr}}_{\mathcal{A},1}(\lambda)$.

   Let us now show these hybrids to be instinguishable.

   - $\mathsf{Hyb}_0 \equiv \mathsf{Hyb}_1$*:* These hybrids are code equivalent, we simply "inlined" the codes of $\mathsf{sPCF.Gen}$ and $\mathsf{sPCF.Eval}$.
   - $\mathsf{Hyb}_1 \overset{c}{\approx} \mathsf{Hyb}_2$*:* These hybrids are indistinguishable by security of ELF. Indeed otherwise, the PPT process of running $\mathsf{fnaPCF.Gen}$, emulating $\mathcal{O}_1$, and running $\mathcal{A}^{\mathcal{O}_1(\cdot)}$ would constitute an efficient distinguisher for the ELF security game.
   - $\mathsf{Hyb}_2 \equiv \mathsf{Hyb}_3$*:* These hybrids are code-equivalent; observe that we simply moved the brunt of the work of $\mathcal{O}_1$ to a pre-processing phase inside Hyb_3.
   - $\mathsf{Hyb}_3 \equiv \mathsf{Hyb}_4$*:* These hybrids are code-equivalent; we simply reorganised the code to introduce $\mathcal{A}_0$ and $\mathcal{A}_1$.
   - $\mathsf{Hyb}_4 \overset{c}{\approx} \mathsf{Hyb}_5$*:* These hybrids are indistinguishable by the property of *non-adaptively pseudorandom $\mathcal{Y}$-correlated outputs* of fnaPCF.

$$\mathsf{Hyb}_3(1^\lambda) \xrightarrow{\text{split adversary}} \mathsf{Hyb}_4(1^\lambda) \xrightarrow{\mathsf{fnaPCF}} \boxed{\mathsf{Hyb}_5(1^\lambda)} \xrightarrow{\text{merge adversary}} \mathsf{Hyb}_6(1^\lambda) \xrightarrow{\text{ELF enum. Im.}}$$

```
$\mathsf{Hyb}_3(1^\lambda)$
$f \leftarrow\!\$\ \mathsf{ELF.Gen}(2^{n(\lambda)}, N(\lambda))$
$(k_0^{\mathsf{fnaPCF}}, k_1^{\mathsf{fnaPCF}}) \leftarrow\!\$\ \mathsf{fnaPCF.Gen}(1^\lambda)$
$\mathcal{Q} \leftarrow \varnothing$
for $z \in \mathsf{Im}(f)$ :
    $y_0 \leftarrow \mathsf{fnaPCF.Eval}(0, k_0^{\mathsf{fnaPCF}}, z)$
    $y_1 \leftarrow \mathsf{fnaPCF.Eval}(1, k_1^{\mathsf{fnaPCF}}, z)$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(z, y_0, y_1)\}$
$b \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}_1(\cdot)}(1^\lambda)$
return $b$

$\mathcal{O}_1(x)$
if $(f(x), y_0, y_1) \in \mathcal{Q}$ :
    return $(y_0, y_1)$
else return $\bot$
```

```
$\mathsf{Hyb}_4(1^\lambda)$  $\boxed{\mathsf{Hyb}_5(1^\lambda)}$
$((x^{(i)})_{i\in[N(\lambda)]}, \mathsf{state}) \leftarrow\!\$\ \mathcal{A}_0(1^\lambda)$
$(k_0^{\mathsf{fnaPCF}}, k_1^{\mathsf{fnaPCF}}) \leftarrow\!\$\ \mathsf{fnaPCF.Gen}(1^\lambda)$
for $i = 1 \ldots N(\lambda)$ :
    $y_0^{(i)} \leftarrow \mathsf{fnaPCF.Eval}(0, k_0^{\mathsf{fnaPCF}}, x^{(i)})$
    $y_1^{(i)} \leftarrow \mathsf{fnaPCF.Eval}(1, k_1^{\mathsf{fnaPCF}}, x^{(i)})$
    $\boxed{(y_0^{(i)}, y_1^{(i)}) \leftarrow\!\$\ \mathcal{Y}}$
$b \leftarrow\!\$\ \mathcal{A}_1(\mathsf{state}, (y_0^{(i)}, y_1^{(i)})_{i\in[N(\lambda)]})$
return $b$

$\mathcal{A}_0(1^\lambda)$
$r_{\mathsf{ELF}} \leftarrow\!\$\ \{0,1\}^\lambda$
$f \leftarrow\!\$\ \mathsf{ELF.Gen}(2^{n(\lambda)}, N(\lambda); r_{\mathsf{ELF}})$
$\mathsf{state} \leftarrow r_{\mathsf{ELF}}$
return $((z)_{z\in\mathsf{Im}(f)}, f)$

$\mathcal{A}_1(\mathsf{state}, (y_0^{(i)}, y_1^{(i)})_{i\in[N(\lambda)]})$
$f \leftarrow\!\$\ \mathsf{ELF.Gen}(2^{n(\lambda)}, N(\lambda); \mathsf{state})$
$(x^{(i)})_{i\in[N(\lambda)]} \leftarrow (z)_{z\in\mathsf{Im}(f)}$
$\mathcal{Q} \leftarrow \{(x^{(i)}, y_0^{(i)}, y_1^{(i)}) : i \in [N(\lambda)]\}$
Define: $\mathcal{O}_1(\cdot) : X \mapsto (Y_0, Y_1)$
        s.t. $(X, Y_0, Y_1) \in \mathcal{Q}$
$b \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}_1(\cdot)}(1^\lambda)$
return $b$
```

```
$\mathsf{Hyb}_6(1^\lambda)$
$f \leftarrow\!\$\ \mathsf{ELF.Gen}(2^{n(\lambda)}, N(\lambda))$
$(k_0^{\mathsf{fnaPCF}}, k_1^{\mathsf{fnaPCF}}) \leftarrow\!\$\ \mathsf{fnaPCF.Gen}(1^\lambda)$
$\mathcal{Q} \leftarrow \varnothing$
for $z \in \mathsf{Im}(f)$ :

    $(y_0, y_1) \leftarrow\!\$\ \mathcal{Y}$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(z, y_0, y_1)\}$
$b \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}_0(\cdot)}(1^\lambda)$
return $b$

$\mathcal{O}_0(x)$
if $(f(x), y_0, y_1) \in \mathcal{Q}$ :
    return $(y_0, y_1)$
else return $\bot$
```

Fig. 19: Sequence of hybrids for proving *strongly pseudorandom $\mathcal{Y}$-correlated outputs* in the proof of Lemma 25 (Part 2/3).

$$\mathsf{Hyb}_7(1^\lambda) \xrightarrow{\text{ELF indist.}} \mathsf{Hyb}_8(1^\lambda) \xrightarrow{\text{inline sPCF}} \mathsf{Hyb}_9(1^\lambda)$$

```
$\mathsf{Hyb}_7(1^\lambda)$
$f \leftarrow\!\$\ \mathsf{ELF.Gen}(2^{n(\lambda)}, N(\lambda))$
$(k_0^{\mathsf{fnaPCF}}, k_1^{\mathsf{fnaPCF}}) \leftarrow\!\$\ \mathsf{fnaPCF.Gen}(1^\lambda)$
$b \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}_0(\cdot)}(1^\lambda)$
return $b$

$\mathcal{O}_0(x)$
if $(x, y_0, y_1) \in \mathcal{Q}$ :
    return $(y_0, y_1)$
else :
    $(y_0, y_1) \leftarrow\!\$\ \mathcal{Y}(1^\lambda)$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(x, y_0, y_1)\}$
    return $(y_0, y_1)$
```

```
$\mathsf{Hyb}_8(1^\lambda)$
$f \leftarrow\!\$\ \mathsf{ELF.Gen}(2^{n(\lambda)}, 2^{n(\lambda)})$
$(k_0^{\mathsf{fnaPCF}}, k_1^{\mathsf{fnaPCF}}) \leftarrow\!\$\ \mathsf{fnaPCF.Gen}(1^\lambda)$
$b \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}_0(\cdot)}(1^\lambda)$
return $b$

$\mathcal{O}_0(x)$
if $(x, y_0, y_1) \in \mathcal{Q}$ :
    return $(y_0, y_1)$
else :
    $(y_0, y_1) \leftarrow\!\$\ \mathcal{Y}(1^\lambda)$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(x, y_0, y_1)\}$
    return $(y_0, y_1)$
```

```
$\mathsf{Hyb}_9(1^\lambda)$
$(k_0, k_1) \leftarrow\!\$\ \mathsf{sPCF.Gen}(1^\lambda)$
$\mathcal{Q} \leftarrow \varnothing$
$b \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}_0(\cdot)}(1^\lambda)$
return $b$

$\mathcal{O}_0(x)$
if $(x, y_0, y_1) \in \mathcal{Q}$ :
    return $(y_0, y_1)$
else :
    $(y_0, y_1) \leftarrow\!\$\ \mathcal{Y}(1^\lambda)$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(x, y_0, y_1)\}$
    return $(y_0, y_1)$
```

Fig. 20: Sequence of hybrids for proving *strongly pseudorandom $\mathcal{Y}$-correlated outputs* in the proof of lemma 25 (Part 3/3). $\mathsf{Hyb}_9 = \mathsf{Exp}^{\mathsf{s\text{-}pr}}_{\mathcal{A},0}(\lambda)$

- $\mathsf{Hyb}_5 \equiv \mathsf{H}_6$*:* These hybrids are code equivalent (this hop is essentially the reverse of the transformation from Hyb_3 to Hyb_4).
- $\mathsf{Hyb}_6 \equiv \mathsf{Hyb}_7$*:* These hybrids are code equivalent (this hop is essentially the reverse of the transformation from Hyb_2 to Hyb_3).
- $\mathsf{Hyb}_7 \overset{c}{\approx} \mathsf{Hyb}_8$*:* These hybrids are indistinguishable by security of ELF (with the exact same argument used to show $\mathsf{Hyb}_1 \overset{c}{\approx} \mathsf{Hyb}_2$).
- $\mathsf{Hyb}_8 \equiv \mathsf{Hyb}_9$*:* These hybrids are code equivalent (this hop is essentially the reverse of the transformation from Hyb_0 to Hyb_1).

2. *Strong PCF security.* Let $\sigma \in \{0,1\}$. Let $\mathcal{A}$ be an non-uniform adversary of size $B(\lambda)$ asking at most $N(\lambda)$ queries to the oracle $\mathcal{O}_b(\cdot)$ (as defined in Figure 10).
   Consider the sequence of hybrids $(\mathsf{Hyb}_i)_{i\in[0,9]}$, as defined in Figures 21 to 23.

```
$\mathsf{Hyb}_0 = \mathsf{Exp}^{\mathsf{s\text{-}sec}}_{\mathcal{A},1,\sigma}(\lambda)$
$(k_0, k_1) \leftarrow\!\$\ \mathsf{sPCF.Gen}(1^\lambda)$

$b \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}_1(\cdot)}(1^\lambda)$
return $b$

$\mathcal{O}_1(x)$
$y_\sigma \leftarrow \mathsf{sPCF.Eval}(\sigma, k_\sigma, x)$
$y_{1-\sigma} \leftarrow \mathsf{Rsample}(1^\lambda, \sigma, y_\sigma)$
return $y_{1-\sigma}$

$\mathsf{sPCF.Gen}(1^\lambda)$
$f \leftarrow\!\$\ \mathsf{ELF.Gen}(2^{n(\lambda)}, 2^{n(\lambda)})$
$(k_0^{\mathsf{fnaPCF}}, k_1^{\mathsf{fnaPCF}}) \leftarrow\!\$\ \mathsf{fnaPCF.Gen}(1^\lambda)$
for $\sigma \in \{0,1\}$ : $k_\sigma \leftarrow (f, k_\sigma^{\mathsf{fnaPCF}})$
return $(k_0, k_1)$

$\mathsf{sPCF.Eval}(\sigma, k_\sigma, x)$
Parse $k_\sigma$ as $k_\sigma = (f, k_\sigma^{\mathsf{fnaPCF}})$
$x' \leftarrow f(x)$
$y_\sigma \leftarrow \mathsf{fnaPCF.Eval}(\sigma, k_\sigma^{\mathsf{fnaPCF}}, x')$
return $y_\sigma$
```

```
$\mathsf{Hyb}_1(1^\lambda)$ $\boxed{\mathsf{Hyb}_2(1^\lambda)}$
$f \leftarrow\!\$\ \mathsf{ELF.Gen}(2^{n(\lambda)}, 2^{n(\lambda)} \boxed{N(\lambda)})$
$(k_0^{\mathsf{fnaPCF}}, k_1^{\mathsf{fnaPCF}}) \leftarrow\!\$\ \mathsf{fnaPCF.Gen}(1^\lambda)$
$b \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}_1(\cdot)}(1^\lambda)$
return $b$

$\mathcal{O}_1(x)$
$y_\sigma \leftarrow \mathsf{fnaPCF.Eval}(\sigma, k_\sigma, x)$
$y_{1-\sigma} \leftarrow \mathsf{Rsample}(1^\lambda, \sigma, y_\sigma)$
return $y_{1-\sigma}$
```

```
$\mathsf{Hyb}_3(\lambda)$
$f \leftarrow\!\$\ \mathsf{ELF.Gen}(2^{n(\lambda)}, N(\lambda))$
$(k_0^{\mathsf{fnaPCF}}, k_1^{\mathsf{fnaPCF}}) \leftarrow\!\$\ \mathsf{fnaPCF.Gen}(1^\lambda)$
$\mathcal{Q} \leftarrow \varnothing$
for $z \in \mathsf{Im}(f)$ :
  $y_\sigma \leftarrow \mathsf{fnaPCF.Eval}(\sigma, k_\sigma^{\mathsf{fnaPCF}}, z)$
  $y_{1-\sigma} \leftarrow \mathsf{Rsample}(1^\lambda, \sigma, y_\sigma)$
  $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(z, y_{1-\sigma})\}$
$b \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}_1(\cdot)}(1^\lambda)$
return $b$

$\mathcal{O}_1(x)$
if $(f(x), y_{1-\sigma}) \in \mathcal{Q}$ :
  return $y_{1-\sigma}$
else :
  return $\bot$
```

Fig. 21: Sequence of hybrids for proving *strong PCF security* in the proof of Lemma 25 (Part 1/3).

Let us now show these hybrids to be instinguishable.

- $\mathsf{Hyb}_0 \equiv \mathsf{Hyb}_1$*:* These hybrids are code equivalent, we simply "inlined" the codes of $\mathsf{sPCF.Gen}$ and $\mathsf{sPCF.Eval}$.
- $\mathsf{Hyb}_1 \overset{c}{\approx} \mathsf{Hyb}_2$*:* These hybrids are indistinguishable by security of ELF. Indeed otherwise, the PPT process of running $\mathsf{fnaPCF.Gen}$, emulating $\mathcal{O}_1$, and running $\mathcal{A}^{\mathcal{O}_1(\cdot)}$ would constitute an efficient distinguisher for the ELF security game.
- $\mathsf{Hyb}_2 \equiv \mathsf{Hyb}_3$*:* These hybrids are code-equivalent; observe that we simply moved the brunt of the work of $\mathcal{O}_1$ to a pre-processing phase inside Hyb_3.
- $\mathsf{Hyb}_3 \equiv \mathsf{Hyb}_4$*:* These hybrids are code-equivalent; we simply reorganised the code to introduce $\mathcal{A}_0$ and $\mathcal{A}_1$.
- $\mathsf{Hyb}_4 \overset{c}{\approx} \mathsf{Hyb}_5$*:* These hybrids are indistinguishable by the property of *fully non-adaptive PCF security* of fnaPCF.
- $\mathsf{Hyb}_5 \equiv \mathsf{Hyb}_6$*:* These hybrids are code equivalent (this hop is essentially the reverse of the transformation from Hyb_3 to Hyb_4).

```
$\mathsf{Hyb}_4(\lambda)$
$((x^{(i)})_{i\in[N(\lambda)]}, \mathsf{state}) \leftarrow\!\!\$\ \mathcal{A}_0(1^\lambda, \sigma)$
$(k_0^{\mathsf{fnaPCF}}, k_1^{\mathsf{fnaPCF}}) \leftarrow\!\!\$\ \mathsf{fnaPCF.Gen}(1^\lambda)$
for $i = 1 \dots N(\lambda)$ :
  $y_\sigma^{(i)} \leftarrow \mathsf{fnaPCF.Eval}(\sigma, k_\sigma^{\mathsf{fnaPCF}}, x^{(i)})$
  $y_{1-\sigma}^{(i)} \leftarrow \mathsf{Rsample}(1^\lambda, \sigma, y_\sigma^{(i)})$
$b \leftarrow\!\!\$\ \mathcal{A}_1(1^\lambda, \sigma, \mathsf{state}, k_\sigma, (x^{(i)}, y_{1-\sigma}^{(i)})_{i\in[N(\lambda)]})$
return $b$

$\mathcal{A}_0(1^\lambda, \sigma)$
$r_{\mathsf{ELF}} \leftarrow\!\!\$\ \{0,1\}^\lambda$
$f \leftarrow\!\!\$\ \mathsf{ELF.Gen}(2^{n(\lambda)}, N(\lambda); r_{\mathsf{ELF}})$
$\mathsf{state} \leftarrow r_{\mathsf{ELF}}$
return $((z)_{z\in\mathsf{Im}(f)}, f)$

$\mathcal{A}_1(1^\lambda, \sigma, \mathsf{state}, k_\sigma, (x^{(i)}, y_{1-\sigma}^{(i)})_{i\in[N(\lambda)]})$
$f \leftarrow\!\!\$\ \mathsf{ELF.Gen}(2^{n(\lambda)}, N(\lambda); \mathsf{state})$
$(x^{(i)})_{i\in[N(\lambda)]} \leftarrow (z)_{z\in\mathsf{Im}(f)}$
$\mathcal{Q} \leftarrow \{(x^{(i)}, y_{1-\sigma}^{(i)}) \colon i \in [N(\lambda)]\}$
Define: $\mathcal{O}_1(\cdot) \colon X \mapsto Y_{1-\sigma}$ s.t. $(X, Y_{1-\sigma}) \in \mathcal{Q}$
$b \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}_1(\cdot)}(1^\lambda)$
return $b$
```

```
$\mathsf{Hyb}_5(\lambda)$
$((x^{(i)})_{i\in[N(\lambda)]}, \mathsf{state}) \leftarrow\!\!\$\ \mathcal{A}_0(1^\lambda, \sigma)$
$(k_0^{\mathsf{fnaPCF}}, k_1^{\mathsf{fnaPCF}}) \leftarrow\!\!\$\ \mathsf{fnaPCF.Gen}(1^\lambda)$
for $i = 1 \dots N(\lambda)$ :
  $y_{1-\sigma}^{(i)} \leftarrow \mathsf{fnaPCF.Eval}(1-\sigma, k_{1-\sigma}^{\mathsf{fnaPCF}}, x^{(i)})$
$b \leftarrow\!\!\$\ \mathcal{A}_1(1^\lambda, \sigma, \mathsf{state}, k_\sigma, (x^{(i)}, y_{1-\sigma}^{(i)})_{i\in[N(\lambda)]})$
return $b$

$\mathcal{A}_0(1^\lambda, \sigma)$
$r_{\mathsf{ELF}} \leftarrow\!\!\$\ \{0,1\}^\lambda$
$f \leftarrow\!\!\$\ \mathsf{ELF.Gen}(2^{n(\lambda)}, N(\lambda); r_{\mathsf{ELF}})$
$\mathsf{state} \leftarrow r_{\mathsf{ELF}}$
return $((z)_{z\in\mathsf{Im}(f)}, f)$

$\mathcal{A}_1(1^\lambda, \sigma, \mathsf{state}, k_\sigma, (x^{(i)}, y_{1-\sigma}^{(i)})_{i\in[N(\lambda)]})$
$f \leftarrow\!\!\$\ \mathsf{ELF.Gen}(2^{n(\lambda)}, N(\lambda); \mathsf{state})$
$(x^{(i)})_{i\in[N(\lambda)]} \leftarrow (z)_{z\in\mathsf{Im}(f)}$
$\mathcal{Q} \leftarrow \{(x^{(i)}, y_{1-\sigma}^{(i)}) \colon i \in [N(\lambda)]\}$
Define: $\mathcal{O}_1(\cdot) \colon X \mapsto Y_{1-\sigma}$ s.t. $(X, Y_{1-\sigma}) \in \mathcal{Q}$
$b \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}_1(\cdot)}(1^\lambda)$
return $b$
```

Fig. 22: Sequence of hybrids for proving *strong PCF security* in the proof of Lemma 25 (Part 2/3).

```
$\mathsf{Hyb}_6(\lambda)$
$f \leftarrow\!\!\$\ \mathsf{ELF.Gen}(2^{n(\lambda)}, N(\lambda))$
$(k_0^{\mathsf{fnaPCF}}, k_1^{\mathsf{fnaPCF}}) \leftarrow\!\!\$\ \mathsf{fnaPCF.Gen}(1^\lambda)$
$\mathcal{Q} \leftarrow \varnothing$
for $z \in \mathsf{Im}(f)$ :
  $y_{1-\sigma}^{(i)} \leftarrow \mathsf{fnaPCF.Eval}(1-\sigma, k_{1-\sigma}^{\mathsf{fnaPCF}}, z)$
  $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(z, y_{1-\sigma})\}$
$b \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}_1(\cdot)}(1^\lambda)$
return $b$

$\mathcal{O}_1(x)$
if $(f(x), y_{1-\sigma}) \in \mathcal{Q}$ :
  return $y_{1-\sigma}$
else return $\perp$
```

```
$\mathsf{Hyb}_7(1^\lambda)\ \boxed{\mathsf{Hyb}_8(1^\lambda)}$
$f \leftarrow\!\!\$\ \mathsf{ELF.Gen}(2^{n(\lambda)}, N(\lambda)\ \boxed{2^{n(\lambda)}})$
$(k_0^{\mathsf{fnaPCF}}, k_1^{\mathsf{fnaPCF}}) \leftarrow\!\!\$\ \mathsf{fnaPCF.Gen}(1^\lambda)$
$b \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}_0(\cdot)}(1^\lambda)$
return $b$

$\mathcal{O}_0(x)$
$y_{1-\sigma} \leftarrow \mathsf{fnaPCF.Eval}(1-\sigma, k_{1-\sigma}, x)$
return $y_{1-\sigma}$
```

```
$\mathsf{Hyb}_9 = \mathsf{Exp}^{\mathsf{s\text{-}sec}}_{\mathcal{A},0,\sigma}(\lambda)$
$(k_0, k_1) \leftarrow\!\!\$\ \mathsf{sPCF.Gen}(1^\lambda)$
$b \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}_0(\cdot)}(1^\lambda)$
return $b$

$\mathcal{O}_0(x)$
$y_\sigma \leftarrow \mathsf{sPCF.Eval}(\sigma, k_\sigma, x)$
$y_{1-\sigma} \leftarrow \mathsf{Rsample}(1^\lambda, \sigma, y_\sigma)$
return $y_{1-\sigma}$
```

Fig. 23: Sequence of hybrids for proving *strong PCF security* in the proof of Lemma 25 (Part 3/3).

- $\mathsf{Hyb}_6 \equiv \mathsf{Hyb}_7$*:* These hybrids are code equivalent (this hop is essentially the reverse of the transformation from Hyb_2 to Hyb_3).
- $\mathsf{Hyb}_7 \overset{c}{\approx} \mathsf{Hyb}_8$*:* These hybrids are indistinguishable by security of ELF (with the exact same argument used to show $\mathsf{Hyb}_1 \overset{c}{\approx} \mathsf{Hyb}_2$).
- $\mathsf{Hyb}_8 \equiv \mathsf{Hyb}_9$*:* These hybrids are code equivalent (this hop is essentially the reverse of the transformation from Hyb_0 to Hyb_1).

□

By combining Lemmas 23 and 25 we immediately get Corollary 26.

Corollary 26 (PI-PCF ∘ naPRF ∘ ELF is a sPCF). *Applying an ELF then a non-adaptive PRF to the input of a pseudorandom-input weak PCF for some correlation $\mathcal{Y}$ yields a strong PCF for the same correlation $\mathcal{Y}$.*

5 Candidate PI-PRFs and PI-PCFs

Our work introduces a strengthening of the wPRF notion, namely, the notion of a PI-PRF. In this section, we overview several wPRF candidates, which are either at the heart of some of the most efficient OPRFs, or the most efficient PCFs known to date. For each of these candidates, we analyze the most natural families of attacks against their security:

- For the wPRF candidate of [BIP$^+$18], which is a wPRF in a very low complexity class, we analyze the candidate in the framework of security against *satistical query algorithms*. Attacks from this framework capture the attacks that invalidated the existence of wPRF (with better than superpolynomial security) in AC^0, and were the main attacks studied in [BIP$^+$18]. We strengthen this result, and show that no statistical query algorithm can break the assumption that the [BIP$^+$18] candidate is a *PI-PRF*.
- For the PCFs, we analyze the two wPRFs at the heart of the two leading PCF candidates in [BCG$^+$20] and [BCG$^+$22]. Since both are LPN-style constructions, we analyze their security against attacks from the *linear test framework*, which captures most known attacks on the LPN assumption and its many variants. For both, we establish win-win results, showing that finding a linear attack against the assumption that these candidates are PI-PRF would have surprising consequences.

Our analyzes provides support to the notion of PI-PRF, showing that natural wPRF candidates used in leading applications are plausibly also wPRF. Furthermore, we note that all of the above candidates are provably *not* strong PRFs.

5.1 Pseudorandom-Input PRF Candidates

In this section, we discuss the assumption that the wPRF candidate of [BIP$^+$18] is also a PI-PRF. To support the security of their candidate, one of the main arguments used by the authors of [BIP$^+$18] is that with high probability over K, the function F_K does not correlate with any fixed sufficiently small function family. This implies that their candidate cannot be broken by *statistical query algorithms* [ABG$^+$14]. More concretely, the candidate of [BIP$^+$18] is of the form $F_K(x) = \mathsf{map}(K \cdot x)$, where K is a matrix, x is a vector, $K \cdot x$ denotes matrix-vector multiplication, and map is some fixed mapping which, on input a vector y, returns $\sum_i [y_i \bmod 2] \bmod 3$. Note that the outputs of this wPRF are over $\{-1, 0, 1\}$. The lack of correlation with any sufficiently small function family is formalized in the following lemma:

Lemma 27 ([BIP$^+$18]). *Let $\mathcal{H} = \{h : \{0,1\}^\lambda \mapsto \{-1,0,1\}\}$ be a collection of functions of size s. Then*

$$\Pr_K\left[\exists h \in \mathcal{H} \;\middle|\; \Pr_x[\mathsf{map}(K \cdot x) = h(x)] > \frac{1}{3} + \frac{1}{2^{\lambda-1}} + \varepsilon\right] \leq \frac{5s}{2^\lambda \cdot \varepsilon^2}.$$

Because Lemma 27 refers to a probability over uniformly random inputs x to the function, it is only meaningful when the function is used as a wPRF. In contrast, we are interested in the setting where the inputs are given by an admissible sampler Sam. Nevertheless, we show that the candidate of [BIP$^+$18] is in fact also immunized against all correlation attacks against its PI-PRF security.

Theorem 28. *Let $\mathcal{H} = \{h : \{0,1\}^\lambda \mapsto \{-1,0,1\}\}$ be a collection of functions of size s. Then there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\Pr_K\left[\exists h \in \mathcal{H} \;\middle|\; \Pr_{r,i}[\mathsf{map}(K \cdot \mathsf{Sam}_i(r)) = h(r,i)] > \frac{1}{3} + \mathsf{negl}(\lambda) + \varepsilon\right] \leq \frac{s}{\varepsilon^2} \cdot \mathsf{negl}(\lambda).$$

In the above theorem, the inner probability is over the random choice of the randomness r of the sampler, and of the index i of the sampler output (*i.e.* is $\mathsf{Sam}(r)$ outputs $(x_1, \cdots, x_q)$, we write $\mathsf{Sam}_i(r)$ for the function that returns x_i). Theorem 28 implies that the PI-PRF security of the candidate of [BIP$^+$18] cannot be broken by statistical query analysis, an important class of attacks against wPRFs. In particular, this captures the attack of Linial, Mansour, and Nisan [LMN89] which breaks all candidates wPRFs in AC^0 in quasipolynomial time.
The term $\mathsf{negl}(\lambda)$ in Theorem 28 directly comes from the negligible bound on the probability that any polynomial-time adversary distinguishes the sampler output from random. Stronger assumptions on the admissible sampler, such as subexponential or exponential pseudorandomness, directly translate to a corresponding smaller $\mathsf{negl}(\lambda)$ term in Theorem 28.

Proof. Let $\mathsf{Sam} = \mathsf{Sam}_{\lambda,p} : \{0,1\}^\ell \mapsto (\{0,1\}^\lambda)^p$ denote an admissible sampler. Fix any $i \leq p$ and let S_i denote $\mathsf{Sam}_i^{-1}(0) = \{r \in \{0,1\}^\ell \;:\; \mathsf{Sam}_i(r) = 0\}$.

Claim. For any $i \leq p$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that $|S_i|/2^\ell = \mathsf{negl}(\lambda)$.

Proof. Assume towards contradiction that there exists $i \leq p$ and a polynomial $q(\lambda)$ such that $|S_i|/2^\ell \geq 1/q(\lambda)$. Let Adv denote the following adversary against the pseudorandomness of Sam: given a tuple $(x_1, \cdots, x_p)$, Adv outputs 1 if $x_i = 0$, and returns a uniformly random bit otherwise. Observe that

$$|\Pr_{r \leftarrow_\$ \{0,1\}^\ell}[1 = \mathcal{A}(\mathsf{Sam}_{\lambda,p}(r))] - \Pr_{\forall i: x_i \leftarrow_\$ \{0,1\}^\lambda}[1 = \mathcal{A}(x_1, \cdots, x_p)]| \geq \left|\frac{1}{q(\lambda)} - \frac{1}{2^\lambda}\right| > \frac{1}{2q(\lambda)},$$

which contradicts the assumption that Sam is an admissible sampler. □

Let $T_M \leftarrow \max_{i \leq p} |\mathsf{Sam}_i^{-1}(0)|$ and $T_m \leftarrow \min_{i \leq p} |\mathsf{Sam}_i^{-1}(0)|$. Note that by the above claim, $T_m/2^\ell$ and $T_M/2^\ell$ are both negligible in λ. The rest of the proof largely follows the analysis of [BIP$^+$18], and adapts it to our setting. Along the way, we also fix a minor bug in the original analysis (we notified the authors). Let $\mathbb{1}(a,b)$ denote the indicator function which outputs 1 iff $a = b$, and 0 otherwise. Fix a single $h \in \mathcal{H}$; then, we will conclude with a union bound over all elements of $\mathcal{H}$. First, we consider the following expectation, which we bound in both directions:

$$\begin{aligned}
&\mathbb{E}_K\left[\Pr_{r,i}[\mathsf{map}(K \cdot \mathsf{Sam}_i(r)) = h(r,i)]\right] = \mathbb{E}_K\left[\mathbb{E}_{r,i}[\mathbb{1}(\mathsf{map}(K \cdot \mathsf{Sam}_i(r)), h(r,i))]\right] \\
=&\mathbb{E}_{r,i}[\mathbb{E}_K[\mathbb{1}(\mathsf{map}(K \cdot \mathsf{Sam}_i(r)), h(r,i))]] \leq \max_i \mathbb{E}_r[\mathbb{E}_K[\mathbb{1}(\mathsf{map}(K \cdot \mathsf{Sam}_i(r)), h(r,i))]] \\
=&\max_i \frac{1}{2^\ell} \cdot \left(\sum_{r:\mathsf{Sam}_i(r)=0} \mathbb{1}(0^\lambda, h(r,i)) + \sum_{r:\mathsf{Sam}_i(r)\neq 0} \mathbb{E}_K[\mathbb{1}(\mathsf{map}(K \cdot \mathsf{Sam}_i(r)), h(r,i))]\right) \\
\leq&\frac{T_M}{2^\ell} + \max_{r,i:\mathsf{Sam}_i(r)\neq 0} \mathbb{E}_K[\mathbb{1}(\mathsf{map}(K \cdot \mathsf{Sam}_i(r)), h(r,i))], \text{ using } \mathbb{1}(0^\lambda, h(r,i)) \leq 1.
\end{aligned}$$

Now, for any fixed r, i such that $\mathsf{Sam}_i(r) \neq 0$, $K \cdot \mathsf{Sam}_i(r)$ is uniformly distributed over $\{0,1\}^\lambda$, independently of $h(r,i)$. As shown in [BIP$^+$18], $1/3 - 1/2^\lambda \leq \mathbb{E}_y[\mathbb{1}(y,b)] \leq 1/3 + 1/2^\lambda$ for any b, hence

$$\mathbb{E}_K\left[\Pr_{r,i}[\mathsf{map}(K \cdot \mathsf{Sam}_i(r)) = h(r,i)]\right] \leq \frac{T_M}{2^\ell} + \frac{1}{3} + \frac{1}{2^\lambda}.$$

In the other direction:

$$\begin{aligned}&\mathbb{E}_K\left[\Pr_{r,i}[\mathsf{map}(K\cdot\mathsf{Sam}_i(r))=h(r,i)]\right]\geq\min_i\mathbb{E}_r[\mathbb{E}_K[\mathbb{1}(\mathsf{map}(K\cdot\mathsf{Sam}_i(r)),h(r,i))]]\\&=\min_i\frac{1}{2^\ell}\cdot\left(\sum_{r:\mathsf{Sam}_i(r)=0}\mathbb{1}(0^\lambda,h(r,i))+\sum_{r:\mathsf{Sam}_i(r)\neq0}\mathbb{E}_K[\mathbb{1}(\mathsf{map}(K\cdot\mathsf{Sam}_i(r)),h(r,i))]\right)\\&\geq\frac{2^\ell-T_m}{2^\ell}\cdot\min_{r,i:\mathsf{Sam}_i(r)\neq0}\mathbb{E}_K[\mathbb{1}(\mathsf{map}(K\cdot\mathsf{Sam}_i(r)),h(r,i))],\text{ using }\mathbb{1}(0^\lambda,h(r,i))\geq0.\end{aligned}$$

Using again the fact that whenever $\mathsf{Sam}_i(r)\neq 0$, $K\cdot\mathsf{Sam}_i(r)$ is uniformly distributed over $\{0,1\}^\lambda$, since $\mathbb{E}_y[\mathbb{1}(y,b)]\geq 1/3-1/2^\lambda$ for any b,

$$\mathbb{E}_K\left[\Pr_{r,i}[\mathsf{map}(K\cdot\mathsf{Sam}_i(r))=h(r,i)]\right]\geq\frac{2^\ell-T_m}{2^\ell}\cdot\left(\frac{1}{3}-\frac{1}{2^\lambda}\right).$$

To finish the proof, as in [BIP+18], we use the Bienaymé-Chebyshev inequality, which states that for any random variable X with finite expected value μ and finite non-zero variance σ^2, for any $k>0$, $\Pr[|X-\mu|>k\sigma]\leq 1/k^2$. This yields

$$\Pr_K\left[\exists h\in\mathcal{H}\;\middle|\;\Pr_{r,i}[\mathsf{map}(K\cdot\mathsf{Sam}_i(r))=h(r,i)]>\frac{1}{3}+\frac{1}{2^\lambda}+\frac{T_M}{2^\ell}+\varepsilon=\frac{1}{3}+\mathsf{negl}(\lambda)+\varepsilon\right]\leq\frac{\sigma^2}{\varepsilon^2},$$

and we need to bound the variance to conclude:

$$\begin{aligned}&\mathbb{E}_K[\mathbb{E}_{r,i}[\mathbb{1}(\mathsf{map}(K\cdot\mathsf{Sam}_i(r)),h(r,i))]^2]\\=&\mathbb{E}_K[\mathbb{E}_{r,i}[\mathbb{1}(\mathsf{map}(K\cdot\mathsf{Sam}_i(r)),h(r,i))]\cdot\mathbb{E}_{r',i'}[\mathbb{1}(\mathsf{map}(K\cdot\mathsf{Sam}_{i'}(r')),h(r',i'))]]\\\leq&\max_{i,i'}\mathbb{E}_{r,r'}[\mathbb{E}_K[\mathbb{1}(\mathsf{map}(K\cdot\mathsf{Sam}_i(r)),h(r,i))]\cdot\mathbb{E}_{r',i'}[\mathbb{1}(\mathsf{map}(K\cdot\mathsf{Sam}_{i'}(r')),h(r',i'))]]\\\leq&\left(\frac{1}{3}+\frac{1}{2^\lambda}\right)^2+\frac{1+2T_M}{2^\ell},\end{aligned}$$

where the last inequality follows from the fact that when $r\neq r'$, $\mathsf{Sam}_i(r)\neq 0$, and $\mathsf{Sam}_{i'}(r')\neq 0$ (which happens for a fraction at least $(1+2T_M)/2^\ell$ of all strings), then $K\cdot\mathsf{Sam}_i(r)$ and $K\cdot\mathsf{Sam}_{i'}(r')$ are uniformly and independently distributed. Eventually, using the definition of the variance,

$$\begin{aligned}\sigma^2&\leq\left(\frac{1}{3}+\frac{1}{2^\lambda}\right)^2+\frac{1+2T_M}{2^\ell}-\left(\frac{2^\ell-T_m}{2^\ell}\cdot\left(\frac{1}{3}-\frac{1}{2^\lambda}\right)\right)^2\\&=\left(\frac{1}{3}+\frac{1}{2^\lambda}\right)^2-\left(\frac{1}{3}-\frac{1}{2^\lambda}\right)^2+\frac{1+2T_M}{2^\ell}+\left(\frac{T_m}{2^\ell}\cdot\left(\frac{1}{3}-\frac{1}{2^\lambda}\right)\right)^2\\&\leq\frac{4}{3\cdot2^\lambda}+\frac{1+2T_M}{2^\ell}+\frac{T_m^2}{9\cdot2^{2\ell}}.\end{aligned}$$

Since $T_M/2^\ell$ and $T_m/2^\ell$ are both negligible in λ, this yields $\sigma^2\leq\mathsf{negl}(\lambda)$. Eventually, we conclude the proof via a straightforward union bound over the s functions $h\in\mathcal{H}$. □

5.2 Implications for Existing PCFs

In this section, we discuss the implications of our result for existing PCF constructions. Currently, there are two main constructions of weak PCFs: a candidate put forth in [BCG+20] (recently refined in [CD23]) and a candidate put forth in [BCG+22]. Both candidates follow a common template, which (at a high level) wraps a particular cryptographic primitive (a function secret-sharing scheme for multipoint functions, or MP-FSS) around a code-based low-complexity wPRF candidate. To apply our compiler, the weak PCF candidates of [BCG+20, BCG+22] must satisfy the stronger PI-PCF security notion, which directly translate to assuming that the underlying wPRF is a PI-PRF. Below, we recall both candidates and show that falsifying the assumption that the underlying wPRF is a PI-PRF would have interesting and surprising consequences.

5.2.1 The two wPRF candidates. Both the candidate of [BCG$^+$20, CD23] and the candidate of [BCG$^+$22] apply the same transform to a base wPRF which reduces to a variant of the learning parity with noise (LPN) assumption. These two assumptions are called respectively *variable-density LPN* and *expand-accumulate LPN.* In the following, $N = 2^D$ is a bound on the maximum number of samples that an adversary can obtain (where $D = D(\lambda)$ is polynomial in the security parameter).

Variable-Density LPN. Fix parameters $\mathsf{par} = (\lambda, D, N = 2^D)$. Let $\mathcal{R}_{\lambda,i}$ be the distribution of random λ-regular vectors over $\mathbb{F}_2^{\lambda\cdot 2^i}$: that is, a sample from $\mathcal{R}_{\lambda,i}$ is obtained by concatenating λ independent length-2^i unit vectors. We let $\mathcal{H}^i_{\mathsf{vd}}(\mathsf{par})$ denote the distribution over $N \times (\lambda \cdot 2^i)$ matrices over $\mathbb{F}_2$ where each row is sampled independently from $\mathcal{R}_{\lambda,i}$, and $\mathcal{H}_{\mathsf{vd}}(\mathsf{par})$ denote the distribution over $\mathbb{F}_2^{N\times 2N}$ obtained by sampling $H_i \leftarrow\$ \mathcal{H}^i_{\mathsf{vd}}(\mathsf{par})$ for $i = 1$ to D and outputting $H = H_1 || \cdots || H_D$. Eventually, we denote by $\mathcal{N}_{\mathsf{vd}}(\mathsf{par})$ the noise distribution obtained by sampling $\vec{e}_i^{\intercal} \leftarrow\$ \mathcal{R}_{\lambda,i}$ and outputting $\vec{e} \leftarrow (\vec{e}_1 /\!/ \cdots /\!/ \vec{e}_D) \in \mathbb{F}_2^{2N}$ (that is, $\vec{e}^{\intercal}$ is distributed as a row of H).

Definition 29 ($\mathsf{VDLPN}(\lambda, D, N)$)**.** *The* variable-density learning parity with noise *assumption with sparsity λ, D blocks, and number of samples N, denoted* $\mathsf{VDLPN}(\lambda, D, N)$*, states that*

$$\{(H, \vec{b}) \mid H \leftarrow\$ \mathcal{H}_{\mathsf{vd}}(\mathsf{par}), \vec{e} \leftarrow\$ \mathcal{N}_{\mathsf{vd}}(\mathsf{par}), \vec{b} \leftarrow H \cdot \vec{e}\} \stackrel{c}{\approx} \{(H, \vec{b}) \mid H \leftarrow\$ \mathcal{H}_{\mathsf{vd}}(\mathsf{par}), \vec{b} \leftarrow\$ \mathbb{F}_2^N\}.$$

The VDLPN assumption parametrized with any $D = \mathsf{poly}(\lambda)$ immediately yields a wPRF F:

- The vector $\vec{e} \leftarrow\$ \mathcal{N}_{\mathsf{vd}}(\mathsf{par})$ defines the secret key of F.
- On input a random $x \leftarrow\$ \{0,1\}^{\lambda\cdot\sum_{i\le D} i}$, parse x into D blocks x_i of $\lambda \cdot i$ bits, each divided into λ strings $x_{i,j} \in \{0,1\}^i$. Map each $x_{i,j}$ to the length-2^i unit vector which has a 1 at $x_{i,j}$. Let $\mathsf{map}(x)$ denote the concatenation of all these unit vectors. Output $F_{\vec{e}}(x) = \mathsf{map}(x)^{\intercal} \cdot \vec{e}$.

For a random x, by construction, $\mathsf{map}(x)$ is equally distributed to sampling a uniformly random column of H. Therefore, breaking the security of the above wPRF after receiving N samples is equivalent to breaking the VDLPN assumption.

Expand-Accumulate LPN. Fix parameters $\mathsf{par} = (\lambda, c, N)$. N is the number of samples, c is a matrix sparsity parameter (typically $c = \Theta(\log N)$ or $\omega(\log N)$), and λ is the Hamming weight of the noise. Let Δ_N denote a $5N$-by-$5N$ lower triangular matrix filled with ones. We let $\mathcal{H}_{\mathsf{ea}}(\mathsf{par})$ denote the distribution obtained by sampling an N-by-$5N$ matrix M whose entries are independent Bernoulli sample equal to 1 with probability $c/2N$, and outputting $H = M \cdot \Delta_N$. We denote by $\mathcal{N}_{\mathsf{ea}}(\mathsf{par})$ the distribution obtained by concatenating t random unit vectors of length N/t.

Definition 30 ($\mathsf{EALPN}(\lambda, c, N)$)**.** *The* expand-accumulate learning parity with noise *assumption with noise weight λ, matrix sparsity c, and number of samples N, denoted* $\mathsf{EALPN}(\lambda, c, N)$*, states that*

$$\{(H, \vec{b}) \mid H \leftarrow\$ \mathcal{H}_{\mathsf{ea}}(\mathsf{par}), \vec{e} \leftarrow\$ \mathcal{N}_{\mathsf{ea}}(\mathsf{par}), \vec{b} \leftarrow H \cdot \vec{e}\} \stackrel{c}{\approx} \{(H, \vec{b}) \mid H \leftarrow\$ \mathcal{H}_{\mathsf{ea}}(\mathsf{par}), \vec{b} \leftarrow\$ \mathbb{F}_2^N\}.$$

5.2.2 Security against linear tests. Both the VDLPN and the EALPN assumptions are recent assumptions, introduced in [BCG$^+$20] and [BCG$^+$22], respectively. To provide support for VDLPN and EALPN, a natural approach is to analyze their security against standard attacks. In the context of LPN variants, the *linear test* framework (which has its roots in the seminal works of Naor and Naor [NN90] and of Mossel, Shpilka, and Trevisan [MST03], first explicitly put forth in [BCG$^+$20] and further used in multiple subsequent works [BCG$^+$21, CRR21, BCG$^+$22, CD23]) provides a unified way to argue security against most standard attacks against LPN (such as Information-Set Decoding (ISD), or Blum-Kalai-Wassermann-style attacks [BKW03], and many more). Concretely, an attack against LPN in the linear test framework proceeds in two stages:

1. First, a matrix H is sampled from the matrix distribution $\mathcal{H}$, and fed to the (unbounded) adversary Adv. The adversary returns a (nonzero) *test vector* $\vec{v} = \mathsf{Adv}(H)$.
2. Second, a noise vector $\vec{e}$ is sampled from the noise distribution $\mathcal{N}$. The *advantage* of the adversary Adv in the linear test game is the bias of the induced distribution $\vec{v} \cdot H \cdot \vec{e}^{\intercal}$.

To formalize this notion, we recall the definition of the bias of a distribution:

Definition 31 (Bias of a Distribution). *Given a distribution $\mathcal{D}$ over $\mathbb{F}^n$ and a vector $\vec{u} \in \mathbb{F}^n$, the bias of $\mathcal{D}$ with respect to $\vec{u}$, denoted* $\mathsf{bias}_{\vec{u}}(\mathcal{D})$*, is equal to*

$$\mathsf{bias}_{\vec{u}}(\mathcal{D}) = \left| \Pr_{\vec{x} \sim \mathcal{D}}[\vec{u} \cdot \vec{x}^{\intercal} = 0] - \Pr_{\vec{x} \sim \mathcal{U}_n}[\vec{u} \cdot \vec{x}^{\intercal} = 0] \right| = \left| \Pr_{\vec{x} \sim \mathcal{D}}[\vec{u} \cdot \vec{x}^{\intercal} = 0] - \frac{1}{|\mathbb{F}|} \right|,$$

where $\mathcal{U}_n$ denotes the uniform distribution over $\mathbb{F}^n$. The bias of $\mathcal{D}$, denoted $\mathsf{bias}(\mathcal{D})$*, is the maximum bias of $\mathcal{D}$ with respect to any nonzero vector $\vec{u}$.*

We say that an instance of the syndrome decoding problem is *secure against linear test* if, with very high probability over the sampling of H in step 1, for any possible adversarial choice of $\vec{v} = \mathsf{Adv}(H)$, the bias of $\vec{v} \cdot H \cdot \vec{e}^{\intercal}$ induced by the random sampling of $\vec{e}$ is negligible. Intuitively, the linear test framework captures any attack where the adversary is restricted to computing a linear function of the syndrome $\vec{b}^{\intercal} = H \cdot \vec{e}^{\intercal}$, but the choice of the linear function itself can depend arbitrarily on the code. Hence, the adversary is restricted in one dimension (it has to be linear in $\vec{b}^{\intercal}$), but can run in unbounded time given H. Then, we say that an LPN-style assumption (ε, δ)*-fools* linear tests if

$$\Pr_H[\mathsf{bias}(\mathcal{D}_H) > \delta] \leq \varepsilon,$$

where $\mathcal{D}_H$ denotes the distribution which samples $\vec{e}$ and outputs the LPN samples $H \cdot \vec{e}$. The following shows that VDLPN cannot be broken by attacks from the linear test framework, which provides strong support for its security:

Theorem 32 ([BCG+20], informal). $\mathsf{VDLPN}(\lambda, D, 2^D)$ *with $D = \Omega(\lambda)$ $(2^{-\Omega(\lambda)}, 2^{-\Omega(\lambda)})$-fools linear tests.*

5.2.3 From security against linear tests to large minimum distance.

A statement regarding security against linear tests is, under the hood, a statement about the minimum distance of a linear code whose parity-check matrix H' is related to H. Below, we make this explicit for VDLPN and EALPN. In the case of VDLPN, it requires a little bit of work to exhibit the right matrix.

For VDLPN. Given matrices $M_1, \cdots, M_n$ (for some n), we let $\mathsf{BD}(M_1, \cdots, M_n)$ denote the block-diagonal matrix whose diagonal blocks are the M_j's. Let $\mathcal{I}_i \in \mathbb{F}_2^{2^i \times 2^D}$ denote the horizontal concatenation of 2^{D-i} identity matrices of size $2^i \times 2^i$ (for any t), and let $B_i \leftarrow \mathsf{BD}(\mathcal{I}_i, \cdots, \mathcal{I}_i)$ (where the number of blocks is equal to λ). We observe that the distribution $\mathcal{N}_{\mathsf{vd}}(\mathsf{par})$ can be equivalently described as follows: sample $\vec{u}$ as the concatenation of $\lambda \cdot D$ length-2^D unit vectors, and output $\vec{e} = \mathsf{BD}(B_1, \cdots, B_D) \cdot \vec{u}$. Note also that $\mathsf{BD}(B_1, \cdots, B_D)$ is a fixed matrix.
Now, sample $H \leftarrow\!\!\$\ \mathcal{H}_{\mathsf{vd}}(\mathsf{par})$ and define $H' \leftarrow H \cdot \mathsf{BD}(B_1, \cdots, B_D)$. The VDLPN assumption is equivalent to the following assumption: given $(H', \vec{b})$, it is hard to distinguish whether $\vec{b}$ is random, or $\vec{b} = H' \cdot \vec{u}$, where $\vec{u}$ is sampled as above. Then, we have the following simple lemma:

Lemma 33. *The code generated by the rows of H' has minimum distance at least $w = \ln(1/\delta)/4 = \Omega(\lambda)$, with probability at least $1 - \varepsilon$ over the choice of H'.*

Proof. The proof is relatively simple. Assume towards contradiction that with probability larger than ε, the code generated by the rows of H' does not have minimum distance w. This means that with probability $\varepsilon' > \varepsilon$, there exists a vector $\vec{v}$ such that $\vec{v}^{\intercal} \cdot H'$ has Hamming weight less than w. Then,

$$\mathsf{bias}_{\vec{v}}(\mathcal{D}_H) = \left| 1/2 - \Pr_{\vec{u}}[\vec{v}^{\intercal} \cdot H' \cdot \vec{u} = 1] \right| \geq \left(1 - \frac{2w}{\lambda D}\right)^{\lambda D} > \delta,$$

which is a contradiction. Above, the bound on the bias follows from the piling-up lemma (which bounds the probability that a XOR of independent samples from a Bernoulli distribution equals 1, with equality when the nonzero entries of $\vec{v}^{\intercal} \cdot H'$ are spread equally among the λD blocks of length 2^D) and the second inequality follows from the standard inequality $(1 - 1/n)^n \geq e^{-1} \cdot (1 - 1/n) > e^{-2}$. □

For EALPN. In the case of EALPN, this is actually much more direct: the matrix H' is simply equal to $H = M \cdot \Delta_N$ (where M is a random sparse matrix and Δ_N a lower triangle of ones). In fact, security against linear test is directly stated as a theorem about the minimum distance of the code spanned by H in [BCG+22]:

Lemma 34 ([BCG+22], Theorem 3.10). *Fix a parameter $c = \omega(\log N)$. The code generated by the rows of $H = M \cdot \Delta_N$ has minimum distance at least $\Omega(N)$, with probability at least $1 - N^{-\omega(1)}$ over the choice of H.*

5.2.4 A win-win result for PI-PRF security against linear tests. Equipped with the above results, we return to our initial question: how plausible is the assumption that the weak PCFs of [BCG+20,CD23] and [BCG+22] are PI-PCFs? As it turns out, this question is equivalent to asking whether the wPRFs defined by VDLPN and EALPN are PI-PRFs. Since the main security argument supporting VDLPN and EALPN is that they are secure against linear tests, it is meaningful to ask whether the corresponding *pseudorandom-input* variants of VDLPN and EALPN resist linear tests, too.

By our above lemmas, this is equivalent to the following problem (we state it for VDLPN for concreteness, but the reasoning is similar for EALPN): given an admissible sampler Sam, if we sample each row h_j of the matrix H as $\mathsf{map}(x_j)^\intercal$, where $(x_1, \cdots, x_N) \leftarrow\$ \mathsf{Sam}$, does $H' = H \cdot \mathsf{BD}(B_1, \cdots, B_D)$ have minimum distance $\Omega(\lambda)$? Let us denote $\mathcal{D}^r$ the distribution of H' when random $x_1, \cdots x_N$ are used, and $\mathcal{D}^{\mathsf{pr}}$ the distribution with $(x_1, \cdots, x_N) \leftarrow\$ \mathsf{Sam}$. Now, because Sam is an admissible sampler, it holds that the distribution of $(x_1, \cdots, x_N)$ is computationally indistinguishable from random. Therefore, $\mathcal{D}^{\mathsf{pr}}$ is computationally indistinguishable from $\mathcal{D}^r$, which samples codes with a minimum distance at least $\Omega(\lambda)$. That is, *no polynomial time adversary can distinguish $H' \leftarrow\$ \mathcal{D}^{\mathsf{pr}}$ from a code with a large minimum distance.* Using the terminology from [BCG+22, Definition 3.12], $\mathcal{D}^{\mathsf{pr}}$ has a large *pseudodistance.*

The existence of codes with a large gap between their pseudodistance and their actual minimum distance is an open problem which has received some attention in the literature. In particular, the hardness of finding a low-weight codeword, when it exists, is equivalent to the *binary SVP assumption* from [AHI+17]. The binary SVP assumption is known to have interesting consequences, such as the existence of collision-resistant hash functions with very low complexity (constant algebraic degree). Therefore, we obtain the following win-win result for PI-PCFs:

Either the VDLPN-based candidate wPRF of [BCG+20, CD23] is also a PI-PRF, or the binary SVP assumption holds with respect to the distribution $\mathcal{D}^{\mathsf{pr}}$.

A similar win-win holds for the EALPN-based wPRF candidate of [BCG+22] as well.

5.2.5 Key-agreement from VDLPN or EALPN. We further note that for the transformations to work, it suffices for the PI-PCF to be pseudorandom with respect to a *specific* admissible sampler Sam. Namely, let the sampler Sam output $(x_1, \cdots, x_N) = \mathsf{PRF}_K(z_1, \cdots, z_N)$, where $(z_1, \cdots, z_N)$ are (non-adaptively) defined by the sampler, and PRF is a pseudorandom function (the key K of the PRF can be sampled randomly by the PCF Gen algorithm, and appended to the PCF keys).

In Appendix B.1, we show analogously to Pietrzak-Sjödin [PS08] that if a wPRF is not also a Sam-PI-PRF, then there exists a key-agreement protocol. Now, let PRF_K be a PRF which is pseudorandom under the VDLPN or EALPN assumption (since they imply one-way functions, they also imply the existence of a PRF). Let us now instante the sampler Sam with PRF_K and assume that the wPRF is not also a Sam-PI-PRF. Then, under the VDLPN or EALPN assumption, the sampler Sam instantiated with PRF_K is an admissible sampler Sam, hence the construction from Appendix B.1 yields a secure key-agreement protocol. Therefore, we get the following win-win result:

Either the VDLPN-based candidate wPRF of [BCG+20, CD23] is also a Sam-PI-PRF, or VDLPN implies key agreement. The same holds for EALPN.

The problem of understanding whether VDLPN implies key agreement was explicitly put forth and studied in [BCG+21]. They showed that some natural approaches which use the Razborov-Smolensky lemma fail to yield key agreement, and could only obtain a positive result under an additional new assumption, called *random LPN is the hardest.*

6 Fujisaki-Okamoto (FO) transform

We now present our modified FO transform as outlined in the technical overview, Section 2. We first define all public-key encryption (PKE) properties that we rely on in this section.

6.1 PKE properties

First, recall that a *Public-Key Encryption* scheme (PKE) consists of three PPT algorithms ($\mathsf{PKE.KGen}$, $\mathsf{PKE.Enc}, \mathsf{PKE.Dec}$) with $\mathsf{PKE.KGen}(1^\lambda) \overset{\$}{\to} (\mathsf{sk}, \mathsf{pk})$, $\mathsf{PKE.Enc}(\mathsf{pk}, m) \overset{\$}{\to} c$ and $\mathsf{PKE.Dec}(\mathsf{sk}, c) \to m'$. We consider PKE that are correct with overwhelming probability over the key sampling, i.e.,

Definition 35. *A* PKE *is correct with overwhelming probability over the key sampling if*

$$\Pr_{(\mathsf{sk},\mathsf{pk}) \leftarrow\$ \mathsf{PKE.KGen}(1^\lambda)}\left[\forall m \in \{0,1\}^\lambda : \mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{PKE.Enc}(\mathsf{pk}, m)) = m\right] = 1 - \mathsf{negl}(\lambda). \quad (3)$$

Recall that the Fujisaki-Okamoto transform boosts indistinguishability under chosen plaintext attacks (IND-CPA) of a PKE to indistinguishability under chosen ciphertext attacks (IND-CCA).

Definition 36 (IND-CPA and IND-CCA secure PKE). *Let* $\mathsf{X} \in \{\mathsf{CPA}, \mathsf{CCA}\}$*.* PKE *is IND-X secure if for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*, the advantage* $\mathsf{Adv}^{\mathsf{X}}_{(\mathcal{A}_0,\mathcal{A}_1),\mathsf{PKE}}(1^\lambda) :=$

$$\left|\Pr\left[1 = \mathsf{Exp}^{\mathsf{X}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),\mathsf{PKE},0}\right] - \Pr\left[1 = \mathsf{Exp}^{\mathsf{X}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),\mathsf{PKE},1}\right]\right| = \mathsf{negl}(\lambda).$$

where the experiment $\mathsf{Exp}^{\mathsf{X}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),\mathsf{PKE},b}$ *are defined below.*

Experiment IND-CPA and IND-CCA Security of a PKE

```
$\mathsf{Exp}^{\mathsf{CPA}\ \mathsf{CCA}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),\mathsf{PKE},0}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{PKE.KGen}(1^n)$
$m, \mathsf{state} \leftarrow\$ \mathcal{A}_0^{\mathsf{DEC}}(\mathsf{pk})$
$c \leftarrow\$ \mathsf{PKE.Enc}(\mathsf{pk}, m)$

$b^* \leftarrow\$ \mathcal{A}_1^{\mathsf{DEC}}(1^n, \mathsf{state}, \mathsf{pk}, c)$
return $b^*$

$\mathsf{DEC}(c^*)$
if $c^* = c$ :
    return $\bot$
return $\mathsf{PKE.Dec}(\mathsf{sk}, c^*)$
```

```
$\mathsf{Exp}^{\mathsf{CPA}\ \mathsf{CCA}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),\mathsf{PKE},1}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{PKE.KGen}(1^n)$
$m, \mathsf{state} \leftarrow\$ \mathcal{A}_0^{\mathsf{DEC}}(\mathsf{pk})$
$c \leftarrow\$ \mathsf{PKE.Enc}(\mathsf{pk}, 0^{|m|})$
    // $c \leftarrow\$ \{0,1\}^{|c|}$ for $-IND-CCA
$b^* \leftarrow\$ \mathcal{A}_1^{\mathsf{DEC}}(1^n, \mathsf{state}, \mathsf{pk}, c)$
return $b^*$

$\mathsf{DEC}(c^*)$
if $c^* = c$ :
    return $\bot$
return $\mathsf{PKE.Dec}(\mathsf{sk}, c^*)$
```

A notion which we put forward in this work and which our modified FO instantiation relies on is that the PKE is invertible when given the randomness.

Definition 37 (Invertibility given Randomness). *A* PKE *is* invertible given randomness *if there exists a PPT algorithm* $\mathcal{I}$ *such that*

$$\Pr_{(\mathsf{sk},\mathsf{pk}) \leftarrow\$ \mathsf{PKE.KGen}(1^\lambda)}\left[\forall r, m \in \{0,1\}^\lambda : \mathcal{I}(\mathsf{pk}, \mathsf{PKE.Enc}(\mathsf{pk}, m; r), r) = m\right] \approx 1 - \mathsf{negl}(\lambda)$$

Remark. Note that Dec receives the *secret-key* sk, while the inverter $\mathcal{I}$ additionally knows randomness r, but, in turn, only gets the *public-key* pk. Instead of $\mathcal{I}(\mathsf{pk}, c, r)$, we write $\mathsf{PKE.Enc}(\mathsf{pk}, \cdot; r)^{-1}(c)$.

6.2 Instantiating modified FO

We first define what it means to instantiate our modified FO transform, then provide our construction and then prove its security.

Definition 38 (Instantiating modified FO). *A PPT sampleable distribution $\mathcal{D}_\mathsf{H}$ instantiates modified* FO*, if*

- *for all IND-CPA secure* PKE *that are invertible given randomness (Definition 37, and*
- *for all 1-time AE-secure* SE *(Definition 43),*

construction $\mathsf{FO}_\mathsf{mod}^{\mathcal{D}_\mathsf{H}}[\mathsf{PKE},\mathsf{SE}]$ *(Fig. 24, middle) is and IND-CCA-secure* PKE *(Definition 36).*

```
FO[PKE, SE]

KGen(1^λ)
  $(\mathsf{sk_{cpa}}, \mathsf{pk_{cpa}}) \leftarrow\$ \mathsf{PKE_{cpa}}(1^\lambda)$
  return $(\mathsf{sk_{cpa}}, \mathsf{pk_{cpa}})$

Enc(pk, m; σ)
  $c_\mathsf{cpa} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \sigma; \mathsf{RO}_1(\sigma||m))$
  $c_\mathsf{sym} \leftarrow \mathsf{SE.Enc}(\mathsf{RO}_2(\sigma), m)$
  return $(c_\mathsf{cpa}, c_\mathsf{sym})$

Dec(sk_cpa, c_cpa, c_sym)
  $\sigma \leftarrow \mathsf{PKE.Dec}(\mathsf{sk_{cpa}}, c_\mathsf{cpa})$
  $k_\mathsf{sym} \leftarrow \mathsf{RO}_2(\sigma)$
  $m \leftarrow \mathsf{SE.Dec}(k_\mathsf{sym}, c_\mathsf{sym})$
  assert $c_\mathsf{cpa} =$
    $\mathsf{PKE.Enc}(\mathsf{pk}, \sigma; \mathsf{RO}_1(\sigma||m))$
  return $m$
```

```
FO_mod^{D_H}[PKE, SE]

KGen(1^λ)
  $(\mathsf{sk_{cpa}}, \mathsf{pk_{cpa}}) \leftarrow\$ \mathsf{PKE_{cpa}}(1^\lambda)$
  $\mathsf{H}_1, \mathsf{H}_2, \mathsf{H}'_1, \mathsf{P} \leftarrow\$ \mathcal{D}_\mathsf{H}(1^\lambda)$
  $\mathsf{pk} \leftarrow (\mathsf{pk_{cpa}}, \mathsf{H}_1, \mathsf{H}_2)$
  $\mathsf{sk} \leftarrow (\mathsf{sk_{cpa}}, \mathsf{H}'_1, \mathsf{P})$
  return $(\mathsf{sk}, \mathsf{pk})$

Enc(pk, m; σ)
  $(\mathsf{pk_{cpa}}, \mathsf{H}_1, \mathsf{H}_2) \leftarrow \mathsf{pk}$
  $c_\mathsf{cpa} \leftarrow \mathsf{PKE_{cpa}.Enc}(\mathsf{pk_{cpa}}, \mathsf{H}_2(\sigma); \mathsf{H}_1(\sigma))$
  $c_\mathsf{sym} \leftarrow \mathsf{SE.Enc}(\mathsf{H}_2(\sigma), m)$
  return $(c_\mathsf{cpa}, c_\mathsf{sym})$

Dec(sk, c_cpa, c_sym)
  $(\mathsf{sk_{cpa}}, \mathsf{H}'_1, \mathsf{P}) \leftarrow \mathsf{sk}$
  $k_\mathsf{sym} \leftarrow \mathsf{PKE_{cpa}.Dec}(\mathsf{sk_{cpa}}, c_\mathsf{cpa})$
  assert $\mathsf{P}(k_\mathsf{sym}) = 1$
  assert $c_\mathsf{cpa} =$
    $\mathsf{PKE_{cpa}.Enc}(\mathsf{pk_{cpa}}, k_\mathsf{sym}; \mathsf{H}'_1(k_\mathsf{sym}))$
  return $\mathsf{SE.Dec}(k_\mathsf{sym}, c_\mathsf{sym})$
```

```
D_H(1^λ)
  $(\mathsf{pk_{cca}}, \mathsf{sk_{cca}}) \leftarrow\$ \mathsf{PKE_{cca}.KGen}(1^\lambda)$
  $w \leftarrow\$ \{0,1\}^\lambda$
  $\mathsf{H}_2 \leftarrow \mathsf{PKE_{cca}.Enc}(\mathsf{pk_{cca}}, w; \cdot)$
  $k_\mathsf{PPRF} \leftarrow\$ \{0,1\}^\lambda$
  $f \leftarrow\$ \mathsf{ELF.Gen}(1^\lambda, 2^\lambda)$
  $\mathsf{H}'_1 \leftarrow \mathsf{PPRF}(k_\mathsf{PPRF}(f(\cdot)))$
  $\mathsf{H}_1 \leftarrow\$ \mathsf{iO}(\mathsf{PPRF}(k_\mathsf{PPRF}(f(\mathsf{H}_2(\cdot)))))$
  $\mathsf{P} \leftarrow (w \overset{?}{=} \mathsf{PKE_{cca}.Dec}(\mathsf{sk_{cca}}, \cdot))$
  return $(\mathsf{H}_1, \mathsf{H}_2, \mathsf{H}'_1, \mathsf{P})$
```

Fig. 24: FO transform and our instantiation of a modified FO transform

Theorem 39 (Instantiation of modified FO). *Assuming that*

- $(\mathsf{PPRF}, \mathsf{Punct})$ *is a puncturable PRF (Definition 40),*
- iO *is an indistinguishability obfuscator (Definition 41),*
- ELF *is an extremely lossy function (Definition 1), and*
- $\mathsf{PKE_{cca}}$ *is \$-IND-CCA-secure* PKE *(Definition 36),*

then distribution $\mathcal{D}_\mathsf{H}$ (Fig. 24, right) instantiates modified FO.

Proof overview. The proof proceeds via 20 hybrids, detailed in Fig. 25-28. We first inline the definition of $\mathsf{FO}_\mathsf{mod}^{\mathcal{D}_\mathsf{H}}[\mathsf{PKE},\mathsf{SE}]$ and the distribution $\mathcal{D}_\mathsf{H}$ in game Hyb_1, then pre-sample randomness for the challenge ciphertext in game Hyb_2.
Hybrids Hyb_3 to Hyb_6 use the Sahai-Waters [SW14] iO+PRG+PPRF trick to remove the randomness of the challenge ciphertext from the circuit H_2 which the adversary knows—recall from Section 2 that, instead of the PRG, we use $\sigma \mapsto \mathsf{PKE_{cca}.Enc}(\mathsf{pk_{cca}}, w; \sigma)$ with a (random, but) fixed message w. As in the iO+PRG+PPRF trick, we then argue that a *random* string of the same length as the ciphertext is, with overwhelming probability not an encryption of w and thus not in the range of the function $\mathsf{Enc}(\mathsf{pk_{cca}}, w; \cdot)$.

Hybrid Hyb_7 replaces the injective ELF key by a suitably lossy ELF key. Note that this step does not affect the randomness of the challenge ciphertext which is uniformly random and independent.

After applying the ELF, in Hyb_8, we can replace decryption by enumerating over all values in the image of the ELF which lead to all possible values that can be legally used as randomness in the IND-CPA-secure PKE. As the IND-CPA-secure PKE is invertible given randomness (Definition 37), the DEC oracle can try out all possible randomness and then run the inverter—and check whether the inverter gave any correct value—there can be at most one by correctness of the PKE (Definition 37).

From Hyb_8 to Hyb_9, we replace the key k_{sym} by 0^{len}, where $\mathsf{len} = |k_{\mathsf{sym}}|$ and can now reduce to IND-CPA security of the PKE since its decryption procedure is not used.

Finally, from Hyb_9 to Hyb_{10}, we replace message m by $0^{|m|}$ and reduce to 1-time AE-security, since k_{sym} is not used anywhere.

Hybrids Hyb_{10} to Hyb_{19} are the analogous inverse steps of hybrids Hyb_0 to Hyb_9.

```
$\boxed{\mathsf{Hyb}_{19}(1^\lambda)} \xleftarrow{\text{reverse inline FO}}$
$\mathsf{Hyb}_0(1^\lambda) \xrightarrow{\text{inline FO}}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{FO}^{\mathcal{D}_\mathsf{H}}_{\mathsf{mod}}[\mathsf{PKE}, \mathsf{SE}].\mathsf{KGen}(1^n)$
$m, \mathsf{state} \leftarrow\$ \mathcal{A}_0^{\mathsf{DEC}}(\mathsf{pk})$
$c \leftarrow\$ \mathsf{FO}^{\mathcal{D}_\mathsf{H}}_{\mathsf{mod}}[\mathsf{PKE}, \mathsf{SE}].\mathsf{Enc}(\mathsf{pk}, m; \boxed{0^{|m|}})$
$b^* \leftarrow\$ \mathcal{A}_1^{\mathsf{DEC}}(1^n, \mathsf{state}, \mathsf{pk}, c)$
return $b^*$

$\mathsf{DEC}(c^*)$
if $c^* = c$ :
    return $\bot$
return $\mathsf{FO}^{\mathcal{D}_\mathsf{H}}_{\mathsf{mod}}[\mathsf{PKE}, \mathsf{SE}].\mathsf{Dec}(\mathsf{sk}, c^*)$
```

Fig. 25: Hybrids Hyb_0 and Hyb_{19}.

$$\begin{tikzcd}
 & \boxed{\mathsf{Hyb}_{18}(1^\lambda)} & \boxed{\mathsf{Hyb}_{17}(1^\lambda)} \arrow[l, "\text{pre-eval+inline hash}"'] & \boxed{\mathsf{Hyb}_{16}(1^\lambda)} \arrow[l, "\text{Punct. cor+iO+inj. } f"'] & \\
{} \arrow[r, "\text{inline FO}"] & \mathsf{Hyb}_1(1^\lambda) \arrow[r, "\text{pre-eval+inline hash}"] & \mathsf{Hyb}_2(1^\lambda) \arrow[r, "\text{Punct. cor+iO+inj. } f"] & \mathsf{Hyb}_3(1^\lambda) \arrow[r, "\text{PPRF+PKE}_{\mathsf{cca}}\text{ cor.}"] & {}
\end{tikzcd}$$

```
$(\mathsf{pk}_{\mathsf{cpa}}, \mathsf{sk}_{\mathsf{cpa}})$
  $\leftarrow\!\!\$\ \mathsf{FO}^{\mathcal{D}_{\mathsf{H}}}_{\mathsf{mod}}[\mathsf{PKE}, \mathsf{SE}].\mathsf{KGen}(1^\lambda)$
$(\mathsf{pk}_{\mathsf{cca}}, \mathsf{sk}_{\mathsf{cca}})$
  $\leftarrow\!\!\$\ \mathsf{PKE}_{\mathsf{cca}}.\mathsf{KGen}(1^\lambda)$
$k_{\mathsf{PPRF}} \leftarrow\!\!\$\ \{0,1\}^\lambda$
$f \leftarrow\!\!\$\ \mathsf{ELF.Gen}(1^\lambda, 2^\lambda)$
$\mathsf{H}_2 \leftarrow \mathsf{PKE}_{\mathsf{cca}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cca}}, w; \cdot)$
$\mathsf{H}_1 \leftarrow\!\!\$\ \mathsf{iO}(C[k_{\mathsf{PPRF}}, f, \mathsf{H}_2])$
$m, \mathsf{state} \leftarrow\!\!\$\ \mathcal{A}_0^{\mathsf{DEC}}((\mathsf{pk}_{\mathsf{cpa}}, \mathsf{H}_1, \mathsf{H}_2)))$
$\sigma \leftarrow\!\!\$\ \{0,1\}^\lambda$
$r_{\mathsf{cpa}} \leftarrow H_1(\sigma);\ k_{\mathsf{sym}} \leftarrow H_2(\sigma)$
$c_{\mathsf{cpa}} \leftarrow \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cpa}}, k_{\mathsf{sym}}; r_{\mathsf{cpa}})$
$c_{\mathsf{sym}} \leftarrow \mathsf{SE.Enc}(k_{\mathsf{sym}}, m\ \boxed{0^{|m|}})$
$b^* \leftarrow\!\!\$\ \mathcal{A}_1^{\mathsf{DEC}}(1^n, \mathsf{state}, (c_{\mathsf{cpa}}, c_{\mathsf{sym}}))$
return $b^*$

$\mathsf{DEC}(c^*_{\mathsf{cpa}}, c^*_{\mathsf{sym}})$
if $(c^*_{\mathsf{cpa}}, c^*_{\mathsf{sym}}) = (c_{\mathsf{cpa}}, c_{\mathsf{sym}})$ :
  return $\perp$
$k^*_{\mathsf{sym}} \leftarrow \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{cpa}}, c^*_{\mathsf{cpa}})$
if $w \neq \mathsf{PKE}_{\mathsf{cca}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{cca}}, k^*_{\mathsf{sym}})$
  return $\perp$
$r^*_{\mathsf{cpa}} \leftarrow \mathsf{PPRF}(k_{\mathsf{PPRF}}, f(k^*_{\mathsf{sym}}))$
$c'_{\mathsf{cpa}} \leftarrow \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cpa}}, k^*_{\mathsf{sym}}; r^*_{\mathsf{cpa}})$
if $c'_{\mathsf{cpa}} \neq c^*_{\mathsf{cpa}}$ :   return $\perp$
return $\mathsf{SE.Dec}(k^*_{\mathsf{sym}}, c^*_{\mathsf{sym}})$

$C[k_{\mathsf{PPRF}}, f, \mathsf{H}_2](\sigma)$
$\mathsf{PPRF}(k_{\mathsf{PPRF}}, f(\mathsf{H}_2(\sigma)))$
```

```
$(\mathsf{pk}_{\mathsf{cpa}}, \mathsf{sk}_{\mathsf{cpa}})$
  $\leftarrow\!\!\$\ \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{KGen}(1^\lambda)$
$(\mathsf{pk}_{\mathsf{cca}}, \mathsf{sk}_{\mathsf{cca}})$
  $\leftarrow\!\!\$\ \mathsf{PKE}_{\mathsf{cca}}.\mathsf{KGen}(1^\lambda)$
$\sigma \leftarrow\!\!\$\ \{0,1\}^\lambda$
$k_{\mathsf{sym}} \leftarrow \mathsf{PKE}_{\mathsf{cca}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cca}}, w; \sigma)$
$k_{\mathsf{PPRF}} \leftarrow\!\!\$\ \{0,1\}^\lambda$
$f \leftarrow\!\!\$\ \mathsf{ELF.Gen}(1^\lambda, 2^\lambda)$
$r_{\mathsf{cpa}} \leftarrow \mathsf{PPRF}(k_{\mathsf{PPRF}}, f(k_{\mathsf{sym}}))$
$\mathsf{H}_2 \leftarrow \mathsf{PKE}_{\mathsf{cca}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cca}}, w; \cdot)$
$\mathsf{H}_1 \leftarrow\!\!\$\ \mathsf{iO}(C[k_{\mathsf{PPRF}}, f, \mathsf{H}_2])$
$m, \mathsf{state} \leftarrow\!\!\$\ \mathcal{A}_0^{\mathsf{DEC}}(\mathsf{pk}_{\mathsf{cpa}}, \mathsf{H}_1, \mathsf{H}_2)$
$c_{\mathsf{cpa}} \leftarrow \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cpa}}, k_{\mathsf{sym}}; r_{\mathsf{cpa}})$
$c_{\mathsf{sym}} \leftarrow \mathsf{SE.Enc}(k_{\mathsf{sym}}, m\ \boxed{0^{|m|}})$
$b^* \leftarrow\!\!\$\ \mathcal{A}_1^{\mathsf{DEC}}(1^n, \mathsf{state}, (c_{\mathsf{cpa}}, c_{\mathsf{sym}}))$
return $b^*$

$\mathsf{DEC}(c^*_{\mathsf{cpa}}, c^*_{\mathsf{sym}})$
if $(c^*_{\mathsf{cpa}}, c^*_{\mathsf{sym}}) = (c_{\mathsf{cpa}}, c_{\mathsf{sym}})$ :
  return $\perp$
if $c^*_{\mathsf{cpa}} = c_{\mathsf{cpa}}$ :
  return $\mathsf{SE.Dec}(k_{\mathsf{sym}}, c^*_{\mathsf{sym}})$
$k^*_{\mathsf{sym}} \leftarrow \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{cpa}}, c^*_{\mathsf{cpa}})$
if $k^*_{\mathsf{sym}} = k_{\mathsf{sym}}$ : return $\perp$
if $w \neq \mathsf{PKE}_{\mathsf{cca}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{cca}}, k^*_{\mathsf{sym}})$
  return $\perp$
$r^*_{\mathsf{cpa}} \leftarrow \mathsf{PPRF}(k_{\mathsf{PPRF}}, f(k^*_{\mathsf{sym}}))$
$c'_{\mathsf{cpa}} \leftarrow \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cpa}}, k^*_{\mathsf{sym}}; r^*_{\mathsf{cpa}})$
if $c'_{\mathsf{cpa}} \neq c^*_{\mathsf{cpa}}$ :   return $\perp$
return $\mathsf{SE.Dec}(k^*_{\mathsf{sym}}, c^*_{\mathsf{sym}})$

$C[k_{\mathsf{PPRF}}, f, \mathsf{H}_2](\sigma)$
$\mathsf{PPRF}(k_{\mathsf{PPRF}}, f(\mathsf{H}_2(\sigma)))$
```

```
$(\mathsf{pk}_{\mathsf{cpa}}, \mathsf{sk}_{\mathsf{cpa}})$
  $\leftarrow\!\!\$\ \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{KGen}(1^\lambda)$
$(\mathsf{pk}_{\mathsf{cca}}, \mathsf{sk}_{\mathsf{cca}})$
  $\leftarrow\!\!\$\ \mathsf{PKE}_{\mathsf{cca}}.\mathsf{KGen}(1^\lambda)$
$\sigma \leftarrow\!\!\$\ \{0,1\}^\lambda$
$k_{\mathsf{sym}} \leftarrow \mathsf{PKE}_{\mathsf{cca}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cca}}, w; \sigma)$
$k_{\mathsf{PPRF}} \leftarrow\!\!\$\ \{0,1\}^\lambda$
$f \leftarrow\!\!\$\ \mathsf{ELF.Gen}(1^\lambda, 2^\lambda)$
$r_{\mathsf{cpa}} \leftarrow \mathsf{PPRF}(k_{\mathsf{PPRF}}, f(k_{\mathsf{sym}}))$
$x^* \leftarrow f(k_{\mathsf{sym}})$
$k^*_{\mathsf{PPRF}} \leftarrow \mathsf{Punct}(k_{\mathsf{PPRF}}, x^*)$
$\mathsf{H}_2 \leftarrow \mathsf{PKE}_{\mathsf{cca}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cca}}, w; \cdot)$
$\mathsf{H}_1 \leftarrow\!\!\$\ \mathsf{iO}(C[k^*_{\mathsf{PPRF}}, x^*, r_{\mathsf{cpa}}, f, \mathsf{H}_2])$
$m, \mathsf{state} \leftarrow\!\!\$\ \mathcal{A}_0^{\mathsf{DEC}}(\mathsf{pk}_{\mathsf{cpa}}, \mathsf{H}_1, \mathsf{H}_2)$
$c_{\mathsf{cpa}} \leftarrow\!\!\$\ \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cpa}}, k_{\mathsf{sym}}; r_{\mathsf{cpa}})$
$c_{\mathsf{sym}} \leftarrow \mathsf{SE.Enc}(k_{\mathsf{sym}}, m\ \boxed{0^{|m|}})$
$b^* \leftarrow\!\!\$\ \mathcal{A}_1^{\mathsf{DEC}}(1^n, \mathsf{state}, (c_{\mathsf{cpa}}, c_{\mathsf{sym}}))$
return $b^*$

$\mathsf{DEC}(c^*_{\mathsf{cpa}}, c^*_{\mathsf{sym}})$
if $(c^*_{\mathsf{cpa}}, c^*_{\mathsf{sym}}) = (c_{\mathsf{cpa}}, c_{\mathsf{sym}})$ :
  return $\perp$
if $c^*_{\mathsf{cpa}} = c_{\mathsf{cpa}}$ :
  return $\mathsf{SE.Dec}(k_{\mathsf{sym}}, c^*_{\mathsf{sym}})$
$k^*_{\mathsf{sym}} \leftarrow \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{cpa}}, c^*_{\mathsf{cpa}})$
if $k^*_{\mathsf{sym}} = k_{\mathsf{sym}}$ : return $\perp$
if $w \neq \mathsf{PKE}_{\mathsf{cca}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{cca}}, k^*_{\mathsf{sym}})$
  return $\perp$
$r^*_{\mathsf{cpa}} \leftarrow \mathsf{PPRF}(k^*_{\mathsf{PPRF}}, f(k^*_{\mathsf{sym}}))$
$c'_{\mathsf{cpa}} \leftarrow \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cpa}}, k^*_{\mathsf{sym}}; r^*_{\mathsf{cpa}})$
if $c'_{\mathsf{cpa}} \neq c^*_{\mathsf{cpa}}$ :   return $\perp$
return $\mathsf{SE.Dec}(k^*_{\mathsf{sym}}, c^*_{\mathsf{sym}})$

$C[k^*_{\mathsf{PPRF}}, x^*, r_{\mathsf{cpa}}, f, \mathsf{H}_2](\sigma)$
if $f(\mathsf{H}_2(\sigma)) = x^*$ :
  return $r_{\mathsf{cpa}}$
else $\mathsf{PPRF}(k^*_{\mathsf{PPRF}}, f(\mathsf{H}_2(\sigma)))$
```

Fig. 26: Hybrids 1-3 and 16-18 for Theorem 39

$$\begin{tikzcd} & \boxed{\mathsf{Hyb}_{15}(1^n)} & & \boxed{\mathsf{Hyb}_{14}(1^n)} \arrow[ll, "\text{\$-IND-CCA}"'] & & \boxed{\mathsf{Hyb}_{13}(1^n)} \arrow[ll, "\text{Punct. cor+iO+inj. } f"'] & \\ {} \arrow[r, "\text{PPRF+PKE}_{\mathsf{cca}}\text{ cor.}"] & \mathsf{Hyb}_4(1^n) \arrow[rr, "\text{\$-IND-CCA}"] & & \mathsf{Hyb}_5(1^n) \arrow[rr, "\text{Punct. cor+iO+inj. } f"] & & \mathsf{Hyb}_6(1^n) \arrow[r, "\text{ELF}"] & {} \end{tikzcd}$$

```
$(\mathsf{pk}_{\mathsf{cpa}}, \mathsf{sk}_{\mathsf{cpa}})$
 $\leftarrow\!\$\ \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{KGen}(1^\lambda)$
$(\mathsf{pk}_{\mathsf{cca}}, \mathsf{sk}_{\mathsf{cca}})$
 $\leftarrow\!\$\ \mathsf{PKE}_{\mathsf{cca}}.\mathsf{KGen}(1^\lambda)$
$\sigma \leftarrow\!\$\ \{0,1\}^\lambda$
$k_{\mathsf{sym}} \leftarrow \mathsf{PKE}_{\mathsf{cca}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cca}}, w; \sigma)$
$k_{\mathsf{PPRF}} \leftarrow\!\$\ \{0,1\}^\lambda$
$f \leftarrow\!\$\ \mathsf{ELF}.\mathsf{Gen}(1^\lambda, 2^\lambda)$
$r_{\mathsf{cpa}} \leftarrow\!\$\ \{0,1\}^\lambda$
$x^* \leftarrow f(k_{\mathsf{sym}})$
$k^*_{\mathsf{PPRF}} \leftarrow \mathsf{Punct}(k_{\mathsf{PPRF}}, x^*)$
$\mathsf{H}_2 \leftarrow \mathsf{PKE}_{\mathsf{cca}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cca}}, w; \cdot)$
$\mathsf{H}_1 \leftarrow\!\$\ \mathsf{iO}(C[k^*_{\mathsf{PPRF}}, x^*, r_{\mathsf{cpa}}, f, \mathsf{H}_2])$
$m, \mathsf{state} \leftarrow\!\$\ \mathcal{A}_0^{\mathsf{DEC}}(\mathsf{pk}_{\mathsf{cpa}}, \mathsf{H}_1, \mathsf{H}_2)$
$c_{\mathsf{cpa}} \leftarrow \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cpa}}, k_{\mathsf{sym}}; r_{\mathsf{cpa}})$
$c_{\mathsf{sym}} \leftarrow \mathsf{SE}.\mathsf{Enc}(k_{\mathsf{sym}}, m\ \boxed{0^{|m|}})$
$b^* \leftarrow\!\$\ \mathcal{A}_1^{\mathsf{DEC}}(1^n, \mathsf{state}, (c_{\mathsf{cpa}}, c_{\mathsf{sym}}))$
return $b^*$

$\mathsf{DEC}(c^*_{\mathsf{cpa}}, c^*_{\mathsf{sym}})$
if $(c^*_{\mathsf{cpa}}, c^*_{\mathsf{sym}}) = (c_{\mathsf{cpa}}, c_{\mathsf{sym}})$ :
  return $\bot$
if $c^*_{\mathsf{cpa}} = c_{\mathsf{cpa}}$ :
  return $\mathsf{SE}.\mathsf{Dec}(k_{\mathsf{sym}}, c^*_{\mathsf{sym}})$
$k^*_{\mathsf{sym}} \leftarrow \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{cpa}}, c^*_{\mathsf{cpa}})$
if $k^*_{\mathsf{sym}} = k_{\mathsf{sym}}$ : return $\bot$
if $w \neq \mathsf{PKE}_{\mathsf{cca}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{cca}}, k^*_{\mathsf{sym}})$
  return $\bot$
$r^*_{\mathsf{cpa}} \leftarrow \mathsf{PPRF}(k^*_{\mathsf{PPRF}}, f(k^*_{\mathsf{sym}}))$
$c'_{\mathsf{cpa}} \leftarrow \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cpa}}, k^*_{\mathsf{sym}}; r^*_{\mathsf{cpa}})$
if $c'_{\mathsf{cpa}} \neq c^*_{\mathsf{cpa}}$ :   return $\bot$
return $\mathsf{SE}.\mathsf{Dec}(k^*_{\mathsf{sym}}, c^*_{\mathsf{sym}})$

$C[k^*_{\mathsf{PPRF}}, x^*, r_{\mathsf{cpa}}, f, \mathsf{H}_2](\sigma)$
if $f(\mathsf{H}_2(\sigma)) = x^*$ :
  return $r_{\mathsf{cpa}}$
else $\mathsf{PPRF}(k^*_{\mathsf{PPRF}}, f(\mathsf{H}_2(\sigma)))$
```

```
$(\mathsf{pk}_{\mathsf{cpa}}, \mathsf{sk}_{\mathsf{cpa}})$
 $\leftarrow\!\$\ \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{KGen}(1^\lambda)$
$(\mathsf{pk}_{\mathsf{cca}}, \mathsf{sk}_{\mathsf{cca}})$
 $\leftarrow\!\$\ \mathsf{PKE}_{\mathsf{cca}}.\mathsf{KGen}(1^\lambda)$

$k_{\mathsf{sym}} \leftarrow\!\$\ \{0,1\}^{\mathsf{len}}$
$k_{\mathsf{PPRF}} \leftarrow\!\$\ \{0,1\}^\lambda$
$f \leftarrow\!\$\ \mathsf{ELF}.\mathsf{Gen}(1^\lambda, 2^\lambda)$
$r_{\mathsf{cpa}} \leftarrow\!\$\ \{0,1\}^\lambda$
$x^* \leftarrow f(k_{\mathsf{sym}})$
$k^*_{\mathsf{PPRF}} \leftarrow \mathsf{Punct}(k_{\mathsf{PPRF}}, x^*)$
$\mathsf{H}_2 \leftarrow \mathsf{PKE}_{\mathsf{cca}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cca}}, w; \cdot)$
$\mathsf{H}_1 \leftarrow\!\$\ \mathsf{iO}(C[k^*_{\mathsf{PPRF}}, x^*, r_{\mathsf{cpa}}, f, \mathsf{H}_2])$
$m, \mathsf{state} \leftarrow\!\$\ \mathcal{A}_0^{\mathsf{DEC}}(\mathsf{pk}_{\mathsf{cpa}}, \mathsf{H}_1, \mathsf{H}_2)$
$c_{\mathsf{cpa}} \leftarrow \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cpa}}, k_{\mathsf{sym}}; r_{\mathsf{cpa}})$
$c_{\mathsf{sym}} \leftarrow \mathsf{SE}.\mathsf{Enc}(k_{\mathsf{sym}}, m\ \boxed{0^{|m|}})$
$b^* \leftarrow\!\$\ \mathcal{A}_1^{\mathsf{DEC}}(1^n, \mathsf{state}, (c_{\mathsf{cpa}}, c_{\mathsf{sym}}))$
return $b^*$

$\mathsf{DEC}(c^*_{\mathsf{cpa}}, c^*_{\mathsf{sym}})$
if $(c^*_{\mathsf{cpa}}, c^*_{\mathsf{sym}}) = (c_{\mathsf{cpa}}, c_{\mathsf{sym}})$ :
  return $\bot$
if $c^*_{\mathsf{cpa}} = c_{\mathsf{cpa}}$ :
  return $\mathsf{SE}.\mathsf{Dec}(k_{\mathsf{sym}}, c^*_{\mathsf{sym}})$
$k^*_{\mathsf{sym}} \leftarrow \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{cpa}}, c^*_{\mathsf{cpa}})$
if $k^*_{\mathsf{sym}} = k_{\mathsf{sym}}$ : return $\bot$
if $w \neq \mathsf{PKE}_{\mathsf{cca}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{cca}}, k^*_{\mathsf{sym}})$
  return $\bot$
$r^*_{\mathsf{cpa}} \leftarrow \mathsf{PPRF}(k^*_{\mathsf{PPRF}}, f(k^*_{\mathsf{sym}}))$
$c'_{\mathsf{cpa}} \leftarrow \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cpa}}, k^*_{\mathsf{sym}}; r^*_{\mathsf{cpa}})$
if $c'_{\mathsf{cpa}} \neq c^*_{\mathsf{cpa}}$ :   return $\bot$
return $\mathsf{SE}.\mathsf{Dec}(k^*_{\mathsf{sym}}, c^*_{\mathsf{sym}})$

$C[k^*_{\mathsf{PPRF}}, x^*, r_{\mathsf{cpa}}, f, \mathsf{H}_2](\sigma)$
if $f(\mathsf{H}_2(\sigma)) = x^*$ :
  return $r_{\mathsf{cpa}}$
else $\mathsf{PPRF}(k^*_{\mathsf{PPRF}}, f(\mathsf{H}_2(\sigma)))$
```

```
$(\mathsf{pk}_{\mathsf{cpa}}, \mathsf{sk}_{\mathsf{cpa}})$
 $\leftarrow\!\$\ \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{KGen}(1^\lambda)$
$(\mathsf{pk}_{\mathsf{cca}}, \mathsf{sk}_{\mathsf{cca}})$
 $\leftarrow\!\$\ \mathsf{PKE}_{\mathsf{cca}}.\mathsf{KGen}(1^\lambda)$

$k_{\mathsf{sym}} \leftarrow\!\$\ \{0,1\}^{\mathsf{len}}$
$k_{\mathsf{PPRF}} \leftarrow\!\$\ \{0,1\}^\lambda$
$f \leftarrow\!\$\ \mathsf{ELF}.\mathsf{Gen}(1^\lambda, 2^\lambda)$



$\mathsf{H}_2 \leftarrow \mathsf{PKE}_{\mathsf{cca}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cca}}, w; \cdot)$
$\mathsf{H}_1 \leftarrow\!\$\ \mathsf{iO}(C[k_{\mathsf{PPRF}}, f, \mathsf{H}_2])$
$m, \mathsf{state} \leftarrow\!\$\ \mathcal{A}_0^{\mathsf{DEC}}(\mathsf{pk}_{\mathsf{cpa}}, \mathsf{H}_1, \mathsf{H}_2)$
$c_{\mathsf{cpa}} \leftarrow\!\$\ \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cpa}}, k_{\mathsf{sym}})$
$c_{\mathsf{sym}} \leftarrow \mathsf{SE}.\mathsf{Enc}(k_{\mathsf{sym}}, m\ \boxed{0^{|m|}})$
$b^* \leftarrow\!\$\ \mathcal{A}_1^{\mathsf{DEC}}(1^n, \mathsf{state}, (c_{\mathsf{cpa}}, c_{\mathsf{sym}}))$
return $b^*$

$\mathsf{DEC}(c^*_{\mathsf{cpa}}, c^*_{\mathsf{sym}})$
if $(c^*_{\mathsf{cpa}}, c^*_{\mathsf{sym}}) = (c_{\mathsf{cpa}}, c_{\mathsf{sym}})$ :
  return $\bot$
if $c^*_{\mathsf{cpa}} = c_{\mathsf{cpa}}$ :
  return $\mathsf{SE}.\mathsf{Dec}(k_{\mathsf{sym}}, c^*_{\mathsf{sym}})$
$k^*_{\mathsf{sym}} \leftarrow \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{cpa}}, c^*_{\mathsf{cpa}})$
if $k^*_{\mathsf{sym}} = k_{\mathsf{sym}}$ : return $\bot$
if $w \neq \mathsf{PKE}_{\mathsf{cca}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{cca}}, k^*_{\mathsf{sym}})$
  return $\bot$
$r^*_{\mathsf{cpa}} \leftarrow \mathsf{PPRF}(k_{\mathsf{PPRF}}, f(k^*_{\mathsf{sym}}))$
$c'_{\mathsf{cpa}} \leftarrow \mathsf{PKE}_{\mathsf{cpa}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{cpa}}, k^*_{\mathsf{sym}}; r^*_{\mathsf{cpa}})$
if $c'_{\mathsf{cpa}} \neq c^*_{\mathsf{cpa}}$ :   return $\bot$
return $\mathsf{SE}.\mathsf{Dec}(k^*_{\mathsf{sym}}, c^*_{\mathsf{sym}})$

$C[k_{\mathsf{PPRF}}, f, \mathsf{H}_2](\sigma)$
$\mathsf{PPRF}(k_{\mathsf{PPRF}}, f(\mathsf{H}_2(\sigma)))$
```

Fig. 27: Hybrids 4-7 and 13-15 for Theorem 39

$$\begin{tikzcd}
{} & \boxed{\mathsf{Hyb}_{12}(1^n)} & \boxed{\mathsf{Hyb}_{11}(1^n)} \arrow[l, "\text{Invertibility}"'] & \boxed{\mathsf{Hyb}_{10}(1^n)} \arrow[l, "\text{IND-CPA}"'] \\
{} \arrow[r, "\text{ELF}"] & \mathsf{Hyb}_7(1^n) \arrow[r, "\text{Invertibility}"] & \mathsf{Hyb}_8(1^n) \arrow[r, "\text{IND-CPA}"] & \mathsf{Hyb}_9(1^n) \arrow[u, "\text{AE}"']
\end{tikzcd}$$

```
$\mathsf{Hyb}_7(1^n)$ / $\mathsf{Hyb}_{12}(1^n)$
$(\mathsf{pk_{cpa}}, \mathsf{sk_{cpa}})$
 $\leftarrow\$ \mathsf{PKE_{cpa}.KGen}(1^\lambda)$
$(\mathsf{pk_{cca}}, \mathsf{sk_{cca}})$
 $\leftarrow\$ \mathsf{PKE_{cca}.KGen}(1^\lambda)$
$k_{\mathsf{sym}} \leftarrow\$ \{0,1\}^{\mathsf{len}}$
$k_{\mathsf{PPRF}} \leftarrow\$ \{0,1\}^\lambda$
$f \leftarrow\$ \mathsf{ELF.Gen}(1^\lambda, \mathsf{lossySize})$
$\mathsf{H}_2 \leftarrow \mathsf{PKE_{cca}.Enc}(\mathsf{pk_{cca}}, w; \cdot)$
$\mathsf{H}_1 \leftarrow\$ \mathsf{iO}(C[k_{\mathsf{PPRF}}, f, \mathsf{H}_2])$
$m, \mathsf{state} \leftarrow\$ \mathcal{A}_0^{\mathsf{DEC}}(\mathsf{pk_{cpa}}, \mathsf{H}_1, \mathsf{H}_2)$
$c_{\mathsf{cpa}} \leftarrow\$ \mathsf{PKE_{cpa}.Enc}(\mathsf{pk_{cpa}}, k_{\mathsf{sym}})$
$c_{\mathsf{sym}} \leftarrow \mathsf{SE.Enc}(k_{\mathsf{sym}}, m \boxed{0^{|m|}})$
$b^* \leftarrow\$ \mathcal{A}_1^{\mathsf{DEC}}(1^n, \mathsf{state}, (c_{\mathsf{cpa}}, c_{\mathsf{sym}}))$
return $b^*$

$\mathsf{DEC}(c^*_{\mathsf{cpa}}, c^*_{\mathsf{sym}})$
if $(c^*_{\mathsf{cpa}}, c^*_{\mathsf{sym}}) = (c_{\mathsf{cpa}}, c_{\mathsf{sym}})$ :
  return $\bot$
if $c^*_{\mathsf{cpa}} = c_{\mathsf{cpa}}$ :
  return $\mathsf{SE.Dec}(k_{\mathsf{sym}}, c^*_{\mathsf{sym}})$
$k^*_{\mathsf{sym}} \leftarrow \mathsf{PKE_{cpa}.Dec}(\mathsf{sk_{cpa}}, c^*_{\mathsf{cpa}})$
if $k^*_{\mathsf{sym}} = k_{\mathsf{sym}}$ : return $\bot$
if $w \neq \mathsf{PKE_{cca}.Dec}(\mathsf{sk_{cca}}, k^*_{\mathsf{sym}})$
  return $\bot$
$r^*_{\mathsf{cpa}} \leftarrow \mathsf{PPRF}(k_{\mathsf{PPRF}}, f(k^*_{\mathsf{sym}}))$
$c'_{\mathsf{cpa}} \leftarrow \mathsf{PKE_{cpa}.Enc}(\mathsf{pk_{cpa}}, k^*_{\mathsf{sym}}; r^*_{\mathsf{cpa}})$
if $c'_{\mathsf{cpa}} \neq c^*_{\mathsf{cpa}}$ :   return $\bot$
return $\mathsf{SE.Dec}(k^*_{\mathsf{sym}}, c^*_{\mathsf{sym}})$

$C[k_{\mathsf{PPRF}}, f, \mathsf{H}_2](\sigma)$
$\mathsf{PPRF}(k_{\mathsf{PPRF}}, f(\mathsf{H}_2(\sigma)))$
```

```
$\mathsf{Hyb}_8(1^n)$ / $\mathsf{Hyb}_{11}(1^n)$
$(\mathsf{pk_{cpa}}, \mathsf{sk_{cpa}})$
 $\leftarrow\$ \mathsf{PKE_{cpa}.KGen}(1^\lambda)$
$(\mathsf{pk_{cca}}, \mathsf{sk_{cca}})$
 $\leftarrow\$ \mathsf{PKE_{cca}.KGen}(1^\lambda)$
$k_{\mathsf{sym}} \leftarrow\$ \{0,1\}^{\mathsf{len}}$
$k_{\mathsf{PPRF}} \leftarrow\$ \{0,1\}^\lambda$
$f \leftarrow\$ \mathsf{ELF.Gen}(1^\lambda, \mathsf{lossySize})$
$\mathsf{H}_2 \leftarrow \mathsf{PKE_{cca}.Enc}(\mathsf{pk_{cca}}, w; \cdot)$
$\mathsf{H}_1 \leftarrow\$ \mathsf{iO}(C[k_{\mathsf{PPRF}}, f, \mathsf{H}_2])$
$m, \mathsf{state} \leftarrow\$ \mathcal{A}_0^{\mathsf{DEC}}(\mathsf{pk_{cpa}}, \mathsf{H}_1, \mathsf{H}_2)$
$c_{\mathsf{cpa}} \leftarrow\$ \mathsf{PKE_{cpa}.Enc}(\mathsf{pk_{cpa}}, k_{\mathsf{sym}})$
$c_{\mathsf{sym}} \leftarrow \mathsf{SE.Enc}(k_{\mathsf{sym}}, m \boxed{0^{|m|}})$
$b^* \leftarrow\$ \mathcal{A}_1^{\mathsf{DEC}}(1^n, \mathsf{state}, (c_{\mathsf{cpa}}, c_{\mathsf{sym}}))$
return $b^*$

$\mathsf{DEC}(c^*_{\mathsf{cpa}}, c^*_{\mathsf{sym}})$
if $(c^*_{\mathsf{cpa}}, c^*_{\mathsf{sym}}) = (c_{\mathsf{cpa}}, c_{\mathsf{sym}})$ :
  return $\bot$
if $c^*_{\mathsf{cpa}} = c_{\mathsf{cpa}}$ :
  return $\mathsf{SE.Dec}(k_{\mathsf{sym}}, c^*_{\mathsf{sym}})$
for $r \in \mathsf{Im}(\mathsf{PPRF}(k_{\mathsf{PPRF}}, f(.)))$ :
  $k \leftarrow \mathsf{PKE_{cpa}.Enc}(\mathsf{pk_{cpa}}, .; r)^{-1}(c^*_{\mathsf{cpa}})$
  if $k \neq \bot$ : $k^*_{\mathsf{sym}} \leftarrow k$
if $k^*_{\mathsf{sym}} = k_{\mathsf{sym}}$ : return $\bot$
if $w \neq \mathsf{PKE_{cca}.Dec}(\mathsf{sk_{cca}}, k^*_{\mathsf{sym}})$
  return $\bot$
$r^*_{\mathsf{cpa}} \leftarrow \mathsf{PPRF}(k_{\mathsf{PPRF}}, f(k^*_{\mathsf{sym}}))$
$c'_{\mathsf{cpa}} \leftarrow \mathsf{PKE_{cpa}.Enc}(\mathsf{pk_{cpa}}, k^*_{\mathsf{sym}}; r^*_{\mathsf{cpa}})$
if $c'_{\mathsf{cpa}} \neq c^*_{\mathsf{cpa}}$ :   return $\bot$
return $\mathsf{SE.Dec}(k^*_{\mathsf{sym}}, c^*_{\mathsf{sym}})$

$C[k_{\mathsf{PPRF}}, f, \mathsf{H}_2](\sigma)$
$\mathsf{PPRF}(k_{\mathsf{PPRF}}, f(\mathsf{H}_2(\sigma)))$
```

```
$\mathsf{Hyb}_9(1^n)$ / $\mathsf{Hyb}_{10}(1^n)$
$(\mathsf{pk_{cpa}}, \mathsf{sk_{cpa}})$
 $\leftarrow\$ \mathsf{PKE_{cpa}.KGen}(1^\lambda)$
$(\mathsf{pk_{cca}}, \mathsf{sk_{cca}})$
 $\leftarrow\$ \mathsf{PKE_{cca}.KGen}(1^\lambda)$
$k_{\mathsf{sym}} \leftarrow\$ \{0,1\}^{\mathsf{len}}$
$k_{\mathsf{PPRF}} \leftarrow\$ \{0,1\}^\lambda$
$f \leftarrow\$ \mathsf{ELF.Gen}(1^\lambda, \mathsf{lossySize})$
$\mathsf{H}_2 \leftarrow \mathsf{PKE_{cca}.Enc}(\mathsf{pk_{cca}}, w; \cdot)$
$\mathsf{H}_1 \leftarrow\$ \mathsf{iO}(C[k_{\mathsf{PPRF}}, f, \mathsf{H}_2])$
$m, \mathsf{state} \leftarrow\$ \mathcal{A}_0^{\mathsf{DEC}}(\mathsf{pk_{cpa}}, \mathsf{H}_1, \mathsf{H}_2)$
$c_{\mathsf{cpa}} \leftarrow\$ \mathsf{PKE_{cpa}.Enc}(\mathsf{pk_{cpa}}, 0^{\mathsf{len}})$
$c_{\mathsf{sym}} \leftarrow \mathsf{SE.Enc}(k_{\mathsf{sym}}, m \boxed{0^{|m|}})$
$b^* \leftarrow\$ \mathcal{A}_1^{\mathsf{DEC}}(1^n, \mathsf{state}, (c_{\mathsf{cpa}}, c_{\mathsf{sym}}))$
return $b^*$

$\mathsf{DEC}(c^*_{\mathsf{cpa}}, c^*_{\mathsf{sym}})$
if $(c^*_{\mathsf{cpa}}, c^*_{\mathsf{sym}}) = (c_{\mathsf{cpa}}, c_{\mathsf{sym}})$ :
  return $\bot$
if $c^*_{\mathsf{cpa}} = c_{\mathsf{cpa}}$ :
  return $\mathsf{SE.Dec}(k_{\mathsf{sym}}, c^*_{\mathsf{sym}})$
for $r \in \mathsf{Im}(\mathsf{PPRF}(k_{\mathsf{PPRF}}, f(.)))$ :
  $k \leftarrow \mathsf{PKE_{cpa}.Enc}(\mathsf{pk_{cpa}}, .; r)^{-1}(c^*_{\mathsf{cpa}})$
  if $k \neq \bot$ : $k^*_{\mathsf{sym}} \leftarrow k$
if $k^*_{\mathsf{sym}} = k_{\mathsf{sym}}$ : return $\bot$
if $w \neq \mathsf{PKE_{cca}.Dec}(\mathsf{sk_{cca}}, k^*_{\mathsf{sym}})$
  return $\bot$
$r^*_{\mathsf{cpa}} \leftarrow \mathsf{PPRF}(k_{\mathsf{PPRF}}, f(k^*_{\mathsf{sym}}))$
$c'_{\mathsf{cpa}} \leftarrow \mathsf{PKE_{cpa}.Enc}(\mathsf{pk_{cpa}}, k^*_{\mathsf{sym}}; r^*_{\mathsf{cpa}})$
if $c'_{\mathsf{cpa}} \neq c^*_{\mathsf{cpa}}$ :   return $\bot$
return $\mathsf{SE.Dec}(k^*_{\mathsf{sym}}, c^*_{\mathsf{sym}})$

$C[k_{\mathsf{PPRF}}, f, \mathsf{H}_2](\sigma)$
$\mathsf{PPRF}(k_{\mathsf{PPRF}}, f(\mathsf{H}_2(\sigma)))$
```

Fig. 28: Hybrids 8-12 for Theorem 39

6.3 Proof of Theorem 39.

We now discuss each of the 19 game-hops provided in Fig. 25-28 in more detail. Hyb 10-19 are described jointly with Hyb 0-9, where message m is replaced by $0^{|m|}$, as is indicated by the box right next to m. Let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ by a PPT adversary with runtime $p(\lambda)$ and integer $c_{\mathcal{A}}$ such that for infinitely many λ, its advantage is greater than $\frac{1}{\lambda^{c_{\mathcal{A}}}}$.

First, note that

$$\begin{aligned}\mathsf{Hyb}_0 &= \mathsf{Exp}^{\mathsf{CCA}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),\mathsf{FO}^{\mathcal{D}_\mathsf{H}}_{\mathsf{mod}}[\mathsf{PKE},\mathsf{SE}],0}\\ \mathsf{Hyb}_{19} &= \mathsf{Exp}^{\mathsf{CCA}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),\mathsf{FO}^{\mathcal{D}_\mathsf{H}}_{\mathsf{mod}}[\mathsf{PKE},\mathsf{SE}],1}\end{aligned} \tag{4}$$

$\mathsf{Hyb}_0 \to \mathsf{Hyb}_1$: Inlining the code of $\mathsf{FO}^{\mathcal{D}_\mathsf{H}}_{\mathsf{mod}}[\mathsf{PKE},\mathsf{SE}]$ and, in particular, $\mathcal{D}_\mathsf{H}$ (cf. Fig. 24). Therefore,

$$\Pr[1 = \mathsf{Hyb}_0] = \Pr[1 = \mathsf{Hyb}_1] \tag{5}$$

$\mathsf{Hyb}_1 \to \mathsf{Hyb}_2$: We inline the code of the two applications of H_1 and H_2, and move the sampling of σ, the computation of k_sym, and the computation of r_cpa further up in the game, since they do not depend on the adversary's output. Moreover, in the DEC oracle, we simplify the case $c^*_\mathsf{cpa} = c_\mathsf{cpa}$ (so it does not use the decryption algorithm) and directly abort when $c^*_\mathsf{cpa} \neq c_\mathsf{cpa}$, but still $k^*_\mathsf{sym} = k_\mathsf{sym}$, since in this case, re-encryption fails. Consequently,

$$\Pr[1 = \mathsf{Hyb}_1] = \Pr[1 = \mathsf{Hyb}_2] \tag{6}$$

$\mathsf{Hyb}_2 \to \mathsf{Hyb}_3$: The change in DEC does not change the behaviour of the DEC oracle, since (i) the case $f(k^*_\mathsf{sym}) = f(k_\mathsf{sym})$ is covered by the prior check whether $k^*_\mathsf{sym} = k_\mathsf{sym}$ due to the injectivity of f and (ii) the case that $f(k^*_\mathsf{sym}) = f(k_\mathsf{sym})$ is covered by puncturing correctness of the PPRF (Definition 40). Similarly, puncturing correctness of the PPRF implies that the circuits $C[k_\mathsf{PPRF}, f, \mathsf{H}_2]$ and $C[k^*_\mathsf{PPRF}, x^*, r_\mathsf{cpa}, f, \mathsf{H}_2](\sigma)$ are functionally equivalent. We reduce to iO security (Definition 41) using the sampler $\mathcal{A}^\mathsf{iO}_0$ which runs the code of Hyb_2 to create $C_0 := C[k_\mathsf{PPRF}, f, \mathsf{H}_2]$ and computes $k^*_\mathsf{PPRF}, x^*, r_\mathsf{cpa}$ as in Hyb_3 to compute $C_1 := C[k^*_\mathsf{PPRF}, x^*, r_\mathsf{cpa}, f, \mathsf{H}_2](\sigma)$, and stores all computed values in state, and $\mathcal{A}^\mathsf{iO}_1$ then simulates Hyb_2 for $\mathcal{A}$, using $\mathsf{H}_1 := \tilde{C}$, where $\tilde{C}$ is the obfuscated circuit received from the iO challenger. Since the DEC oracles are functionally equivalent, $\mathcal{A}^\mathsf{iO}_1$ perfectly simulates Hyb_2 when $\tilde{C} = \mathsf{iO}(C_0)$ and perfectly simulates Hyb_3 when $\tilde{C} = \mathsf{iO}(C_1)$. We obtain

$$|\Pr[1 = \mathsf{Hyb}_2] - \Pr[1 = \mathsf{Hyb}_3]| \leq \mathsf{Adv}^{\mathsf{iO}}_{\mathcal{A}^\mathsf{iO}_0,\mathcal{A}^\mathsf{iO}_1} \tag{7}$$

$\mathsf{Hyb}_3 \to \mathsf{Hyb}_4$: We reduce to PPRF security (Definition 40) where $\mathcal{A}^\mathsf{PPRF}_0$ samples x uniformly at random and simulates Hyb_3 around it. We obtain

$$|\Pr[1 = \mathsf{Hyb}_3] = \Pr[1 = \mathsf{Hyb}_4]| \leq \mathsf{Adv}^{\mathsf{PPRF}}_{\mathcal{A}^\mathsf{PPRF}_0,\mathcal{A}^\mathsf{PPRF}_1} \tag{8}$$

$\mathsf{Hyb}_4 \to \mathsf{Hyb}_5$: We reduce to \$-IND-CCA security of $\mathsf{PKE}_\mathsf{cca}$ (Definition 36) where $\mathcal{A}^\mathsf{CCA}_0$ embeds the public-key it receives as pk_cca and forwards all decryption calls to its DEC oracle, and embeds its challenge ciphertext as a challenge ciphertext in the game it simulates. Note that k_sym is the challenge ciphertext of $\mathsf{PKE}_\mathsf{cca}$ and the check whether $k^*_\mathsf{sym} = k_\mathsf{sym}$ prevents the challenge ciphertext k_sym from being submitted to the DEC oracle of $\mathcal{A}^\mathsf{CCA}_0$ and $\mathcal{A}^\mathsf{CCA}_1$. We obtain

$$|\Pr[1 = \mathsf{Hyb}_4] = \Pr[1 = \mathsf{Hyb}_5]| \leq \mathsf{Adv}^{\mathsf{\$\text{-}CCA}}_{(\mathcal{A}^\mathsf{CCA}_0,\mathcal{A}^\mathsf{CCA}_1),\mathsf{PKE}}(1^\lambda) \tag{9}$$

$\mathsf{Hyb}_5 \to \mathsf{Hyb}_6$: We construct $(\mathcal{B}^\mathsf{iO}_0, \mathcal{B}^\mathsf{iO}_1)$ analogously to the game-hop from Hyb_2 to Hyb_3 and obtain

$$|\Pr[1 = \mathsf{Hyb}_5] = \Pr[1 = \mathsf{Hyb}_6]| \leq \mathsf{Adv}^{\mathsf{iO}}_{\mathcal{B}^\mathsf{iO}_0,\mathcal{B}^\mathsf{iO}_1} \tag{10}$$

$\mathsf{Hyb}_6 \to \mathsf{Hyb}_7$: We choose the ELF parameters adequately and obtain a reduction $\mathcal{A}^\mathsf{ELF}$ with

$$|\Pr[1 = \mathsf{Hyb}_6] = \Pr[1 = \mathsf{Hyb}_7]| \leq \mathsf{Adv}^{\mathsf{ELF}}_{\mathcal{A}^\mathsf{ELF}} \tag{11}$$

$\mathsf{Hyb}_7 \to \mathsf{Hyb}_8$: Invertibility given randomness (Definition 37) and correctness of $\mathsf{PKE}_\mathsf{cpa}$ imply that *if* c^*_cpa is in the image of $\mathsf{PKE}_\mathsf{cpa}(\mathsf{pk}_\mathsf{cpa}, k^*_\mathsf{sym}, \mathsf{PPRF}(k_\mathsf{PPRF}, f(.)))$ for some k^*_sym, then (i) k^*_sym is unique by correctness and (ii) the inverter finds k^*_sym by invertibility given randomness.

Moreover, the DEC oracle returns $\perp$ for everything outside the image of $\mathsf{PKE}_{\mathsf{cpa}}(\mathsf{pk}_{\mathsf{cpa}}, ., \mathsf{PPRF}(k_{\mathsf{PPRF}}, f(.)))$ so it suffices to search through these values. Hence, in this step, we only lose the correctness parameter and obtain

$$|\Pr[1 = \mathsf{Hyb}_7] = \Pr[1 = \mathsf{Hyb}_8]| \approx \mathsf{negl}(\lambda) \tag{12}$$

$\mathsf{Hyb}_8 \to \mathsf{Hyb}_9$: We reduce to IND-CPA security of $\mathsf{PKE}_{\mathsf{cpa}}$ (Definition 36) constructing an adversary $(\mathcal{A}_0^{\mathsf{CPA}}, \mathcal{A}_1^{\mathsf{CPA}})$ which embeds the pk it receives as $\mathsf{pk}_{\mathsf{cpa}}$ and its challenge ciphertext as c_{cpa}. We obtain

$$|\Pr[1 = \mathsf{Hyb}_8] = \Pr[1 = \mathsf{Hyb}_9]| \leq \mathsf{Adv}^{\mathsf{CPA}}_{(\mathcal{A}_0^{\mathsf{CPA}}, \mathcal{A}_1^{\mathsf{CPA}}), \mathsf{PKE}}(1^\lambda) \tag{13}$$

$\mathsf{Hyb}_9 \to \mathsf{Hyb}_{10}$: We reduce to 1-time AE security of SE (Definition 42) where $\mathcal{A}_0^{\mathsf{SE}}$ uses m for its challenge query. We obtain

$$|\Pr[1 = \mathsf{Hyb}_9] = \Pr[1 = \mathsf{Hyb}_{10}]| \leq \mathsf{Adv}^{\mathsf{AE}}_{(\mathcal{A}_0^{\mathsf{AE}}, \mathcal{A}_1^{\mathsf{AE}}), \mathsf{SE}}(1^\lambda) \tag{14}$$

$\mathsf{Hyb}_{10} \to \mathsf{Hyb}_{19}$: Analogous to the corresponding game-hops, with the same reductions which replace m by $0^{|m|}$. We denote the corresponding adversaries by an additional superscript rev for *rev*erse direction.

Putting Equations (5)-(14) together, we obtain that one of the advantages must be non-negligible or the ELF advantage must be non-negligibly bigger than $\frac{1}{\lambda^\alpha}$ (cf. Definition 1) for infinitely many security parameters.

References

ABG⁺14. Adi Akavia, Andrej Bogdanov, Siyao Guo, Akshay Kamath, and Alon Rosen. Candidate weak pseudorandom functions in AC^0 o MOD_2. In Moni Naor, editor, *ITCS 2014: 5th Conference on Innovations in Theoretical Computer Science*, pages 251–260, Princeton, NJ, USA, January 12–14, 2014. Association for Computing Machinery.

AHI⁺17. Benny Applebaum, Naama Haramaty, Yuval Ishai, Eyal Kushilevitz, and Vinod Vaikuntanathan. Low-complexity cryptographic hash functions. In Christos H. Papadimitriou, editor, *ITCS 2017: 8th Innovations in Theoretical Computer Science Conference*, volume 4266, pages 7:1–7:31, Berkeley, CA, USA, January 9–11, 2017. LIPIcs.

BCG⁺19. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 489–518, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

BCG⁺20. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Correlated pseudorandom functions from variable-density LPN. In *61st Annual Symposium on Foundations of Computer Science*, pages 1069–1080, Durham, NC, USA, November 16–19, 2020. IEEE Computer Society Press.

BCG⁺21. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Low-complexity weak pseudorandom functions in `AC0[MOD2]`. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part IV*, volume 12828 of *Lecture Notes in Computer Science*, pages 487–516, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany.

BCG⁺22. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. Correlated pseudorandomness from expand-accumulate codes. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 603–633, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany.

BCH⁺22. Nir Bitansky, Arka Rai Choudhuri, Justin Holmgren, Chethan Kamath, Alex Lombardi, Omer Paneth, and Ron D. Rothblum. PPAD is as hard as LWE and iterated squaring. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022: 20th Theory of Cryptography Conference, Part II*, volume 13748 of *Lecture Notes in Computer Science*, pages 593–622, Chicago, IL, USA, November 7–10, 2022. Springer, Heidelberg, Germany.

BDG⁺13. Nir Bitansky, Dana Dachman-Soled, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, Adriana López-Alt, and Daniel Wichs. Why "Fiat-Shamir for proofs" lacks a proof. In Amit Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 182–201, Tokyo, Japan, March 3–6, 2013. Springer, Heidelberg, Germany.

Bea92. Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432, Santa Barbara, CA, USA, August 11–15, 1992. Springer, Heidelberg, Germany.

Bea96. Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th Annual ACM Symposium on Theory of Computing*, pages 479–488, Philadephia, PA, USA, May 22–24, 1996. ACM Press.

BFM15. Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Random-oracle uninstantiability from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 428–455, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

BHK13. Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 398–415, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

BHR12. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 134–153, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany.

BIP$^+$18. Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu. Exploring crypto dark matter: New simple PRF candidates and their applications. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 699–729, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany.

BKW03. Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003.

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

BRS03. John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In Kaisa Nyberg and Howard M. Heys, editors, *SAC 2002: 9th Annual International Workshop on Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 62–75, St. John's, Newfoundland, Canada, August 15–16, 2003. Springer, Heidelberg, Germany.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press.

CD23. Geoffroy Couteau and Clément Ducros. Pseudorandom correlation functions from variable-density LPN, revisited. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023: 26th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 13941 of *Lecture Notes in Computer Science*, pages 221–250, Atlanta, GA, USA, May 7–10, 2023. Springer, Heidelberg, Germany.

CGH98. Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218, Dallas, TX, USA, May 23–26, 1998. ACM Press.

CHL22a. Sílvia Casacuberta, Julia Hesse, and Anja Lehmann. Sok: Oblivious pseudorandom functions. In *7th IEEE European Symposium on Security and Privacy, EuroS&P 2022, Genoa, Italy, June 6-10, 2022*, pages 625–646. IEEE, 2022.

CHL22b. Sílvia Casacuberta, Julia Hesse, and Anja Lehmann. Sok: Oblivious pseudorandom functions. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 625–646, 2022.

CKU20. Geoffroy Couteau, Shuichi Katsumata, and Bogdan Ursu. Non-interactive zero-knowledge in pairing-free groups from weaker assumptions. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 442–471, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany.

CRR21. Geoffroy Couteau, Peter Rindal, and Srinivasan Raghuraman. Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 502–534, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany.

DGH$^+$21. Itai Dinur, Steven Goldfeder, Tzipora Halevi, Yuval Ishai, Mahimna Kelkar, Vivek Sharma, and Greg Zaverucha. MPC-friendly symmetric cryptography from alternating moduli: Candidates, protocols, and applications. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology –*

CRYPTO 2021, Part IV, volume 12828 of *Lecture Notes in Computer Science*, pages 517–547, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany.

FIPR05. Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 303–324, Cambridge, MA, USA, February 10–12, 2005. Springer, Heidelberg, Germany.

FO13. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, January 2013.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany.

GGM84. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th Annual Symposium on Foundations of Computer Science*, pages 464–479, Singer Island, Florida, October 24–26, 1984. IEEE Computer Society Press.

GHM$^+$21. Craig Gentry, Shai Halevi, Bernardo Magri, Jesper Buus Nielsen, and Sophia Yakoubov. Random-index PIR and applications. In Kobbi Nissim and Brent Waters, editors, *TCC 2021: 19th Theory of Cryptography Conference, Part III*, volume 13044 of *Lecture Notes in Computer Science*, pages 32–61, Raleigh, NC, USA, November 8–11, 2021. Springer, Heidelberg, Germany.

GK03. Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th Annual Symposium on Foundations of Computer Science*, pages 102–115, Cambridge, MA, USA, October 11–14, 2003. IEEE Computer Society Press.

HL08. Carmit Hazay and Yehuda Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 155–175, San Francisco, CA, USA, March 19–21, 2008. Springer, Heidelberg, Germany.

IKNP03. Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.

JKKZ21. Ruta Jawale, Yael Tauman Kalai, Dakshita Khurana, and Rachel Yun Zhang. SNARGs for bounded depth computations and PPAD hardness from sub-exponential LWE. In Samir Khuller and Virginia Vassilevska Williams, editors, *53rd Annual ACM Symposium on Theory of Computing*, pages 708–721, Virtual Event, Italy, June 21–25, 2021. ACM Press.

JKR19. Stanislaw Jarecki, Hugo Krawczyk, and Jason K. Resch. Updatable oblivious key management for storage systems. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 379–393, London, UK, November 11–15, 2019. ACM Press.

LMN89. Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, Fourier transform, and learnability. In *30th Annual Symposium on Foundations of Computer Science*, pages 574–579, Research Triangle Park, NC, USA, October 30 – November 1, 1989. IEEE Computer Society Press.

MOZ22. Alice Murphy, Adam O'Neill, and Mohammad Zaheri. Instantiability of classical random-oracle-model encryption transforms. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part IV*, volume 13794 of *Lecture Notes in Computer Science*, pages 323–352, Taipei, Taiwan, December 5–9, 2022. Springer, Heidelberg, Germany.

MST03. Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On e-biased generators in NC0. In *44th Annual Symposium on Foundations of Computer Science*, pages 136–145, Cambridge, MA, USA, October 11–14, 2003. IEEE Computer Society Press.

NIS16. NIST. Post-quantum cryptography standardisation. 2016. `https://csrc.nist.gov/projects/post-quantum-cryptography`.

NN90. Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. In *22nd Annual ACM Symposium on Theory of Computing*, pages 213–223, Baltimore, MD, USA, May 14–16, 1990. ACM Press.

NR95. Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. In *36th Annual Symposium on Foundations of Computer Science*, pages 170–181, Milwaukee, Wisconsin, October 23–25, 1995. IEEE Computer Society Press.

PS08. Krzysztof Pietrzak and Johan Sjödin. Weak pseudorandom functions in minicrypt. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008: 35th International Colloquium on Automata, Languages and Programming, Part II*, volume 5126 of *Lecture Notes in Computer Science*, pages 423–436, Reykjavik, Iceland, July 7–11, 2008. Springer, Heidelberg, Germany.

Rab05. Michael O. Rabin. How to exchange secrets with oblivious transfer. Cryptology ePrint Archive, Report 2005/187, 2005. `https://eprint.iacr.org/2005/187`.

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

Zha16. Mark Zhandry. The magic of ELFs. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 479–508, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

Zha19. Mark Zhandry. On ELFs, deterministic encryption, and correlated-input security. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 3–32, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany.

A Additional Preliminaries

A.1 Puncturable PRF

A puncturable PRF is a PRF with the additional feature that the PRF key k can be punctured at a point x so that (1) the punctured key k^* allows to compute the PRF everwhere except for x and (2) the PRF value at point x is pseudorandom *even when given* k^*. We now state the definition formally.

Definition 40 (Puncturable PRF). *A puncturable PRF (PPRF) consists of two algorithms* PPRF *and* Punct *such that* PPRF *is a strong PRF and such that the following are satisfied:*

Puncturing correctness. *For all key $k \in \{0,1\}^\lambda$, all $x, x' \in \{0,1\}^\lambda$ with $x \neq x'$,*

$$\mathsf{PPRF}(k, x') = \mathsf{PPRF}(\mathsf{Punct}(k, x), x').$$

Security. *For all PPT samplers $\mathcal{A}_0$ and PPT adversaries $\mathcal{A}_1$, the advantage $\mathsf{Adv}^{\mathsf{PPRF}}_{\mathcal{A}_0,\mathcal{A}_1}(1^\lambda) :=$*

$$\begin{aligned}&|\Pr_{k \leftarrow\$\{0,1\}^\lambda,(x,\mathsf{state})\leftarrow\$\mathcal{A}_0(1^\lambda),k^*\leftarrow\$\mathsf{Punct}(k,x),y\leftarrow\mathsf{PPRF}(k,x)}[1 = \mathcal{A}_1(x, k^*, y, \mathsf{state})]\\ &- \Pr_{k \leftarrow\$\{0,1\}^\lambda,(x,\mathsf{state})\leftarrow\$\mathcal{A}_0(1^\lambda),k^*\leftarrow\$\mathsf{Punct}(k,x),y\leftarrow\$\{0,1\}^\lambda}[1 = \mathcal{A}_1(x, k^*, y, \mathsf{state})]|\end{aligned}$$

is negligible.

A.2 Indistinguishability Obfuscation

Definition 41. *A PPT algorithm* iO *is an indistinguishability obfuscator if it satisfies the following two properties for all integers c.*

Correctness. *For all circuits C with input length λ,*

$$\Pr_{x\leftarrow\$\{0,1\}^\lambda,\tilde{C}\leftarrow\$\mathsf{iO}(C,1^\lambda,c)}\left[C(x) = \tilde{C}(x)\right] = 1$$

Security. *For all PPT samplers $\mathcal{A}_0$ with*

$$\Pr_{(C_0,C_1,1^c,\mathsf{state})\leftarrow\$\mathcal{A}_0(1^\lambda)}\left[\max\{|C_0|, |C_1|\} \leq \lambda^c \wedge \forall x \in \{0,1\}^\lambda C_0(x) = C_1(x)\right] = 1 - \mathsf{negl}(\lambda)$$

and all PPT algorithms $\mathcal{A}_1$, it holds that the advantage $\mathsf{Adv}^{\mathsf{iO}}_{\mathcal{A}_0,\mathcal{A}_1} :=$

$$\begin{aligned}&|\Pr_{(C_0,C_1,1^c,\mathsf{state})\leftarrow\$\mathcal{A}_0(1^\lambda),\tilde{C}\leftarrow\$\mathsf{iO}(C_0,1^c,1^\lambda)}\left[1 = \mathcal{A}_1(\tilde{C}, \mathsf{state})\right]\\ &- \Pr_{(C_0,C_1,1^c,\mathsf{state})\leftarrow\$\mathcal{A}_0(1^\lambda),\tilde{C}\leftarrow\$\mathsf{iO}(C_1,1^c,1^\lambda)}\left[1 = \mathcal{A}_1(\tilde{C}, \mathsf{state})\right]| = \mathsf{negl}(\lambda).\end{aligned}$$

We omit 1^λ and 1^c from the input of the indistinguishability obfuscator for conciseness.

A.3 Symmetric Encryption

Definition 42 (Deterministic symmetric-key encryption scheme (SE)). *A* deterministic symmetric-key encryption *scheme (*SE*) consists of two PPT algorithms* $(\mathsf{SE.Enc}, \mathsf{SE.Dec})$ *such that*

Encryption $\mathsf{SE.Enc}(k, m) \to c$
Decryption $\mathsf{SE.Dec}(k, c) \to m'$

and such that it is correct with high probability over the key sampling, i.e.,

$$\Pr_{k \leftarrow\$ \{0,1\}^\lambda, c \leftarrow\$ \mathsf{SE.Enc}(k,m)}[\forall m \in \{0,1\}^* : \mathsf{SE.Dec}(k, c) = m] = 1 - \mathsf{negl}(\lambda).$$

Definition 43 (AE secure SE). *A symmetric encryption scheme* SE *is 1-time.AE secure if for all PPT adversaries* $\mathcal{A} =$

$$\left|\Pr\left[1 = \mathsf{Exp}^{\mathsf{AE}}_{\mathcal{A},\mathsf{SE},0}\right] - \Pr\left[1 = \mathsf{Exp}^{\mathsf{AE}}_{\mathcal{A},\mathsf{SE},1}\right]\right| = \mathsf{negl}(\lambda).$$

where the experiments $\mathsf{Exp}^{\mathsf{AE}}_{\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1),\mathsf{SE},b}$ *are defined below. For cleaner presentation we assume w.l.o.g. that* $\mathcal{A}$ *never submits ciphertexts received from* ENC *to the decryption oracle* DEC.

Experiment 1-time AE Security of a SE

```
$\mathsf{Exp}^{\mathsf{AE}}_{\mathcal{A},\mathsf{SE},0}$
$k \leftarrow\$ \mathsf{PKE.KGen}(1^n)$
$b^* \leftarrow\$ \mathcal{A}^{\mathsf{ENC},\mathsf{DEC}}(1^n)$
return $b^*$

$\mathsf{ENC}(m)$
if $c = \perp$
    $c \leftarrow \mathsf{SE.Enc}(k, m)$
return $c$

$\mathsf{DEC}(c^*)$
if $c \neq c^*$ :
    return $\mathsf{SE.Dec}(k, c^*)$
return $\perp$
```

```
$\mathsf{Exp}^{\mathsf{AE}}_{\mathcal{A},\mathsf{SE},1}$
$k \leftarrow\$ \mathsf{SE.KGen}(1^n)$
$b^* \leftarrow\$ \mathcal{A}^{\mathsf{ENC},\mathsf{DEC}}(1^n)$
return $b^*$

$\mathsf{ENC}(m)$
if $c = \perp$
    $c \leftarrow \mathsf{SE.Enc}(k, 0^{|m|})$
return $c$

$\mathsf{DEC}(c^*)$
return $\perp$
```

B Win-Win-Results: infinitely often key agreement (io-KA)

B.1 wPRF not PI-PRF implies io-KA

Proof of Theorem 4. Let wPRF be a weak PRF. If wPRF is *not* a pseudorandom-input PRF, then there exists an admissible sampler $\mathsf{Sam}_{\lambda,p}$ and a $\mathsf{PPT}\mathcal{A}$ with advantage ϵ in the security game of fig. 1d (as instantiated with wPRF as the "candidate PI-PRF"). Consider the protocol of Figure 29 (parameterised by $\mathcal{A}, \mathsf{Sam}_{\lambda,p}, \mathsf{wPRF}$), which we will now show to be a $(\frac{1}{2}+\epsilon)$-correct single-bit infinitely often key-agreement protocol.

Protocol Infinitely Often Key Agreement

```
Alice                                                   Bob
                                                        $k \leftarrow\!\$ \{0,1\}^\lambda$
$r \leftarrow\!\$ \{0,1\}^{\mathrm{inLen}(\mathsf{Sam}_{\lambda,p})}$          $b \leftarrow\!\$ \{0,1\}$
$\vec{x} \leftarrow \mathsf{Sam}_{\lambda,p}(r)$        $\xrightarrow{\vec{x}}$
                                                        if $b = 0$ then
                                                          for $x_i \in \vec{x}$ do
                                                            $y_i \leftarrow f(k, x_i)$
                                                        else
                                                          for $x_i \in \vec{x}$ do
                                                            $y_i \leftarrow\!\$ \{0,1\}^\lambda$
                                    $\xleftarrow{\vec{y}}$
$b \leftarrow \mathcal{A}(\vec{x}, \vec{y}, r)$
return $b$                                              return $b$
```

Fig. 29: Infinitely often key-agreement protocol, assuming the existence of a weak PRF which is not a pseudorandom-input PRF.

Correctness, *i.e.* the fact that Alice and Bob output the same bit with probability at least $(\frac{1}{2} + \epsilon)$ follows immediately from the success probability of $\mathcal{A}$ in breaking the security game of Figure 1d. As for security, assume, for contradiction, that there is an eavesdropper $\mathcal{B}(\vec{x}, \vec{y})$ that can guess b with probability $1/2 + \mu$ for some non-negligible μ, given only the transcript of the protocol, i.e. $(\vec{x}, \vec{y})$. We reach a contradiction by considering the following game hops:

- **Game 0:** Above protocol with $b = 0$
- **Game 1:** Same as Game 0, except $\vec{x}$ is sampled uniformly at random, instead of using Sam
- **Game 2:** Same as Game 1, except $b = 1$ (and hence the protocol samples $\vec{y}$ at random.)
- **Game 3:** Above protocol with $b = 1$. (Same as Game 2 but using Sam for sampling $\vec{x}$.)

Now $\mathcal{B}$ must be able to distinguish a pair of consecutive games. However:
Game 0 is indistinguishable from Game 1 by admissibility of Sam. Game 1 is indistinguishable from Game 2 by wPRF security of f. Game 2 is indistinguishable from Game 3 by admissibility of Sam. So we reach a contradiction, so such $\mathcal{B}$ cannot exist. □

B.2 wPCF not PI-PCF implies io-KA

Proof of Theorem 21. Let wPCF be a weak PCF for some correlation $\mathcal{Y}$. If wPCF is *not* a pseudorandom-input PCF for $\mathcal{Y}$, then at least one of the following properties is not true: *pseudorandom-input pseudorandom $\mathcal{Y}$-correlated outputs* or *pseudorandom-input PCF security.* One of the following statements is therefore true:

- There exists a non-uniform polytime adversary $\mathcal{A}^{\mathsf{pr}}$ and a non-negligible function $\epsilon(\cdot)$, such that for infinitely many $\lambda \in \mathbb{N}$, there exists a polynomial N and an admissible sampler $\mathsf{Sam}_{n(\lambda),N(\lambda)}$ such that:
$$|\Pr[\mathsf{Exp}^{\mathsf{PI\text{-}pr}}_{\mathcal{A}^{\mathsf{pr}},N,0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{PI\text{-}pr}}_{\mathcal{A}^{\mathsf{pr}},N,1}(\lambda) = 1]| > \epsilon(\lambda) \qquad (15)$$
where $\mathsf{Exp}^{\mathsf{PI\text{-}pr}}_{\mathcal{A}^{\mathsf{pr}},N,b}$ $(b \in \{0,1\})$ is defined as in fig. 11 (but parameterised by the PCF $\mathsf{wPCF} = (\mathsf{wPCF.Gen}, \mathsf{wPCF.Eval})$).

- There exists $\sigma \in \{0,1\}$ and a non-uniform polytime adversary $\mathcal{A}^{\mathsf{sec}}_{\sigma}$ and a non-negligible function $\epsilon(\cdot)$, such that for infinitely many $\lambda \in \mathbb{N}$, there exists a polynomial N and an admissible sampler $\mathsf{Sam}_{n(\lambda),N(\lambda)}$
$$|\Pr[\mathsf{Exp}^{\mathsf{PI\text{-}sec}}_{\mathcal{A}^{\mathsf{sec}}_{\sigma},N,\sigma,0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{PI\text{-}sec}}_{\mathcal{A}^{\mathsf{sec}}_{\sigma},N,\sigma,1}(\lambda) = 1]| > \epsilon(\lambda) \tag{16}$$
where $\mathsf{Exp}^{\mathsf{PI\text{-}sec}}_{\mathcal{A}^{\mathsf{sec}}_{\sigma},N,\sigma,b}$ ($b \in \{0,1\}$) is defined as in Figure 12 (but parameterised by the PCF $\mathsf{wPCF} = (\mathsf{wPCF.Gen}, \mathsf{wPCF.Eval})$).

In either case, there is an infinitely often key-agreement protocol with correctness $\frac{1}{2} + \epsilon$.
We now show that regardless which proposition holds there exists an infinitely often key-agreement protocol with correctness $\frac{1}{2} + \epsilon$.

- *Given the existence of $\mathcal{A}^{\mathsf{pr}}$.* Consider the protocol of fig. 30. By eq. (15), Alice and Bob will output the same bit with probability at least $\frac{1}{2} + \epsilon$ for infinitely many values of the security parameter, hence infinitely often correctness (recall that ϵ is non-negligible). For security consider an eavesdropper Eve with access to the transcript of the communication between Alice and Bob. Let $\lambda \in \mathbb{N}$ be a security parameter. Because the sampler $\mathsf{Sam}_{n(\lambda),N(\lambda)}$ is admissible, Eve cannot distinguish between the transcript of the real protocol, and that of a variant where Alice samples the $(x^{(i)})_{i\in[N(\lambda)]}$ uniformly at random. In that variant however, Eve's advantage in guessing b cannot be better than negligible, because the outputs of wPCF (on uniformly random inputs $(x^{(i)})_{i\in[N(\lambda)]}$) are pseudorandomly $\mathcal{Y}$-correlated. Hence security of the io-KA protocol.

Protocol Infinitely Often Key Agreement, given $\mathcal{A}^{\mathsf{pr}}$

Alice		**Bob**
$r \leftarrow\$ \{0,1\}^{\lambda}$		$(k_0, k_1) \leftarrow\$ \mathsf{wPCF.Gen}(1^{\lambda})$ $b \leftarrow\$ \{0,1\}$
$(x^{(i)})_{i\in[N(\lambda)]} \leftarrow \mathsf{Sam}_{n(\lambda),N(\lambda)}(r)$	$\xrightarrow{(x^{(i)})_{i\in[N(\lambda)]}}$	

		if $b = 0$ **then**
		for $i \in [N(\lambda)]$ **do**
		$(y_0^{(i)}, y_1^{(i)}) \leftarrow \mathcal{Y}(1^{\lambda})$
		else
		for $i \in [N(\lambda)]$ **do**
		for $\sigma \in \{0,1\}$ **do**
		$y_{\sigma}^{(i)} \leftarrow \mathsf{wPCF.Eval}(\sigma, k_{\sigma}, x_{\sigma}^{(i)})$

	$\xleftarrow{(y_0^{(i)}, y_1^{(i)})_{i\in[N(\lambda)]}}$	
$b \leftarrow \mathcal{A}^{\mathsf{pr}}(1^{\lambda}, (x^{(i)}, y_0^{(i)}, y_1^{(i)})_{i\in[N(\lambda)]}, r)$ **return** b		**return** b

Fig. 30: Infinitely often key-agreement scheme, assuming the existence of a wPCF which does not satisfy definition 19.

- *Given the existence of $\mathcal{A}^{\mathsf{sec}}_{\sigma}$, for some $\sigma \in \{0,1\}$.* Consider the protocol of fig. 31. By eq. (16), Alice and Bob will output the same bit with probability at least $\frac{1}{2} + \epsilon$ for infinitely many values of the security parameter, hence infinitely often correctness (recall that ϵ is non-negligible). For security consider an eavesdropper Eve with access to the transcript of the communication between Alice and Bob. Let $\lambda \in \mathbb{N}$ be a security parameter. Because the sampler $\mathsf{Sam}_{n(\lambda),N(\lambda)}$ is admissible, Eve cannot distinguish between the transcript of the real protocol, and that of a variant where Alice samples the $(x^{(i)})_{i\in[N(\lambda)]}$ uniformly at random. In that variant however, Eve's advantage in guessing b cannot be better than negligible, by weak PCF security of wPCF. Hence security of the io-KA protocol.

Protocol Infinitely Often Key Agreement, given $\mathcal{A}^{\mathsf{sec}}_{\sigma}$

```
Alice                                                              Bob

$r \leftarrow\$ \{0,1\}^{\lambda}$                                  $(k_0, k_1) \leftarrow\$ \mathsf{wPCF.Gen}(1^{\lambda})$
                              $\xleftarrow{k_\sigma}$                $b \leftarrow\$ \{0,1\}$
$(x^{(i)})_{i \in [N(\lambda)]} \leftarrow \mathsf{Sam}_{n(\lambda),N(\lambda)}(r)$
                              $\xrightarrow{(x^{(i)})_{i \in [N(\lambda)]}}$
                                                                   if $b = 0$ then
                                                                     for $i \in [N(\lambda)]$ do
                                                                       $y^{(i)}_{\sigma} \leftarrow \mathsf{wPCF.Eval}(\sigma, k_\sigma, x^{(i)})$
                                                                       $y^{(i)}_{1-\sigma} \leftarrow \mathsf{RSample}(1^{\lambda}, \sigma, k_\sigma)$
                                                                   else
                                                                     for $i \in [N(\lambda)]$ do
                                                                       $y^{(i)}_{1-\sigma} \leftarrow \mathsf{wPCF.Eval}(1-\sigma, k_{1-\sigma}, x^{(i)})$
                              $\xleftarrow{(y^{(i)}_{1-\sigma})_{i \in [N(\lambda)]}}$
$b \leftarrow \mathcal{A}^{\mathsf{sec}}_{\sigma}(1^{\lambda}, \sigma, k_\sigma,$
        $(x^{(i)}, y^{(i)}_{1-\sigma})_{i \in [N(\lambda)]}, r)$
return $b$                                                         return $b$
```

Fig. 31: Infinitely often key-agreement scheme, assuming the existence of a wPCF which does not satisfy definition 20.

In either case, there exists an infinitely often key-agreement scheme. □