Fast Public-Key Silent OT and More from Constrained Naor-Reingold

Dung Bui, Geoffroy Couteau, Pierre Meyer, Alain Passelègue, Mahshid Riahinia

Fast Public-Key Silent OT and More from Constrained Naor-Reingold

Dung Bui[1], Geoffroy Couteau[1], Pierre Meyer[2], Alain Passelègue[3,4], and Mahshid Riahinia[4]

[1] Université Paris Cité, CNRS, IRIF, FRANCE. {bui,couteau}@irif.fr
[2] Aarhus Universitet, DENMARK. pierre.meyer@cs.au.dk
[3] CryptoLab Inc., FRANCE. alain.passelegue@cryptolab.co.kr
[4] ENS de Lyon, Laboratoire LIP (U. Lyon, CNRS, ENSL, Inria, UCBL), FRANCE. mahshid.riahinia@ens-lyon.fr

Abstract. Pseudorandom Correlation Functions (PCFs) allow two parties, given correlated evaluation keys, to locally generate arbitrarily many pseudorandom correlated strings, e.g. Oblivious Transfer (OT) correlations, which can then be used by the two parties to jointly run secure computation protocols.

In this work, we provide a novel and simple approach for constructing PCFs for OT correlation, by relying on constrained pseudorandom functions for a class of constraints containing a weak pseudorandom function (wPRF). We then show that tweaking the Naor-Reingold pseudorandom function and relying on low-complexity pseudorandom functions allow us to instantiate our paradigm. We further extend our ideas to obtain efficient *public-key* PCFs, which allow the distribution of correlated keys between parties to be non-interactive: each party can generate a pair of public/secret keys, and any pair of parties can locally derive their correlated evaluation key by combining their secret key with the other party's public key.

In addition to these theoretical contributions, we detail various optimizations and provide concrete instantiations of our paradigm relying on the Boneh-Ishai-Passelègue-Sahai-Wu wPRF and the Goldreich-Applebaum-Raykov wPRF. Putting everything together, we obtain public-key PCFs with a throughput of 15k-40k OT/s, which is of a similar order of magnitude to the state-of-the-art *interactive* PCFs and about 4 orders of magnitude faster than state-of-the art *public-key* PCFs.

As a side result, we also show that public-key PCFs can serve as a building block to construct reusable designated-verifier non-interactive zero-knowledge proofs (DV-NIZK) for NP. Combined with our instantiations, this yields simple and efficient reusable DV-NIZKs for NP in pairing-free groups.

Table of Contents

Fast Public-Key Silent OT and More from Constrained Naor-Reingold 1
Dung Bui, Geoffroy Couteau, Pierre Meyer, Alain Passelègue, and Mahshid Riahinia
1 Introduction 3
2 Technical Overview 5
2.1 A PCF for OT from Pseudorandomly Constrained PRFs 5
2.2 A CPRF for Inner-Product Membership from the Naor-Reingold PRF 6
2.3 Inner-Product Membership Weak Pseudorandom Functions 8
2.4 Optimizations 10
2.5 Final PCF Construction 11
2.6 Concrete Parameters 12
2.7 Public Key PCF 14
2.8 Application: A simple reusable DV-NIZK reusable 16
3 Preliminaries 17
3.1 Pseudorandom Functions 17
3.2 Constrained Pseudorandom Functions 17
3.3 Reverse-Sampleable Correlations 18
3.4 Pseudorandom Correlation Functions 19
3.5 NIZKs 22
3.6 Variants of Power-DDH 24
3.7 Decision Composite Residuosity Assumption 24
3.8 Pedersen Commitment Scheme 25
4 Constraining the Naor-Reingold PRF 25
4.1 Inner Product Membership CPRF from Naor-Reingold 25
4.2 Compressing the keys 27
4.3 Application: A Puncturable PRF in NC^1 29
5 Fast PCFs for OTs from Pseudorandomly Constrained PRFs 31
5.1 General Template 31
5.2 Pseudorandom Constraints Expressed as IPM 37
5.3 Candidate IPM-wPRF 38
5.4 Distributed Interactive Key Generation via MPC 40
6 Public-Key PCF for OT Correlations 42
6.1 Formal Definition 42
6.2 A Public-Key PCF via Bellare-Micali Non-Interactive OT 44
6.3 A Better Construction from Paillier-ElGamal 45
6.4 Reducing The Public Keys Size to $\mathcal{O}(n^{2/3})$ 51
7 DV-NIZKs from PK-PCFs 53
7.1 Construction of reusable DVNIZK 53
7.2 Concrete Instantiation 58

1 Introduction

Efficient procedures to generate correlated randomness are at the heart of modern secure computation. Starting with the seminal work of Beaver [Bea95], many protocols achieving impressive performances have been designed in a model where the parties are given access to a trusted source of correlated randomness [NNOB12, DPSZ12, FKOS15, LPSY15, DNNR17, WRK17, HSS20, DILO22]. As an example, $O(n)$ instances of a random oblivious transfer suffice to evaluate any size-n circuit using the seminal GMW protocol [GMW87], using as little as four bits of communication per AND gate.

Due to the efficiency of protocols in the correlated randomness model, a popular paradigm in secure computation is to divide the protocol into two phases: in the *preprocessing* phase, which is independent of the inputs (and can be executed ahead of time), long correlated random strings are securely generated using a dedicated protocol. Then, in the *online* phase, this correlated randomness is consumed by a fast and lightweight protocol. Traditional approaches for generating the correlated randomness were based on oblivious transfer extension [IKNP03] or somewhat homomorphic encryption [DPSZ12]. They incur a large $\Omega(\lambda \cdot n)$ communication overhead for n-gate circuits and typically form the main efficiency bottleneck of the protocol.

Generating pseudorandom correlations. Recently, a new paradigm has emerged which enables the *silent* generation of long correlated *pseudo*random strings [BCGI18, BCG+19b, BCG+19a], removing essentially all of the communication in the preprocessing phase. Concretely, this is made possible by the mean of cryptographic primitives, such as *pseudorandom correlation generators* (PCG) [BCG+19b] and *pseudorandom correlated functions* (PCFs) [BCG+20].

A PCG is a pair of algorithms $(\mathsf{PCG.Gen}, \mathsf{PCG.Expand})$ where $\mathsf{PCG.Gen}$ produces two short keys $(\mathsf{k}_0, \mathsf{k}_1)$, and $\mathsf{PCG.Expand}(\sigma, \mathsf{k}_\sigma)$ produces a long string y_σ such that (y_0, y_1) form pseudorandom samples from the target correlation. PCGs enable silent secure computation as follows: using a small distributed protocol to securely generate the keys $(\mathsf{k}_0, \mathsf{k}_1)$, two parties can afterwards locally expand them into long correlated pseudorandom strings without any further communication. The online phase proceeds as before.

PCGs suffer from a considerable limitation: after distributing the keys, the parties are bound to generate *all at once* a priori *fixed amount* of correlated randomness. PCFs overcome this issue: a PCF is a pair of algorithms $(\mathsf{PCF.Gen}, \mathsf{PCF.Eval})$ where $\mathsf{PCF.Gen}$ produces two short keys $(\mathsf{k}_0, \mathsf{k}_1)$, and $\mathsf{PCF.Eval}(\sigma, \mathsf{k}_\sigma, x)$ outputs y_σ^x where for each new input x, (y_0^x, y_1^x) appears like a fresh sample from the target correlation. Hence, after distributively generating the keys $(\mathsf{k}_0, \mathsf{k}_1)$ once and for all, two parties can generate on-the-fly any amount of target correlations in all their future secure computations.

The line of work on PCGs and PCFs has been fairly successful: modern PCG protocols for the oblivious transfer (OT) correlation (often called *silent OT extension*) can stretch up to 10M OT/s on one core of a standard laptop [CRR21, BCG+22, RRT23] from keys in the 10∼20kB range, and the fastest PCFs for OT [BCG+22] can generate up to 100k OT/s on one core of a standard laptop.

Public-key silent OT. The silent generation of correlated randomness from PCGs or PCFs requires two parties to engage in an interactive protocol to securely generate the PCG/PCF keys. *Public-key* PCFs reduce this interactive phase to a bare minimum, by replacing it with a public-key setup. More precisely, after publishing their public keys online, any pair of parties on a network can start generating correlated randomness, without *any interaction* beyond the initial PKI. Public-key silent correlated randomness generation is somewhat of a holy grail in this line of work: it would represent a major step towards bridging the usability gap between secure communication (since PKI suffices to enable efficient pairwise secure communication) and secure computation, but public-key PCFs for OTs have so far proven considerably harder to achieve than standard PCG and PCFs. Until recently, we simply had no public key silent OT construction, beyond heavy-hammer constructions from obfuscation or threshold multikey FHE.

This changed recently with the result of [OSY21], which achieved the first practical public-key silent OT, assuming the quadratic residuosity assumption and the existence of correlation-robust hash functions. However, the efficiency of the new construction of [OSY21] still lags way behind that of state-of-the-art PCFs for OTs. Concretely, their construction relies on a new distributed discrete logarithm protocol that allows two parties, given multiplicative shares of a value G^x (where G generates a suitable DLog-easy group), to non-interactively compute additive shares of x. The

public-key silent OT construction of [OSY21] has public keys of size around 1kB for one of the parties, and about 50kB for the other. In terms of computational efficiency, the cost of generating a single OT correlation is dominated by λ exponentiations with an exponent in $\mathbb{Z}_{N\cdot 2^\lambda}$, where N is an RSA modulus. Using $\lambda = 128$ and $\log N = 3072$, this translates to 128 exponentiations with 3200-bit exponents and takes about one second on one core of a standard laptop, which is between four and five orders of magnitude slower than the state-of-the-art PCF of [BCG+22]. In summary, as of today, the fundamental goal of obtaining concretely efficient and usable public-key silent OTs remains wide open.

Our Results. As our main contribution, we construct the first concretely efficient public-key PCF for oblivious transfers. Our approach departs significantly from all previous works, and we obtain several additional contributions. Our new public-key PCF features public key sizes in the 30kB range, and the cost of generating each OT is dominated by *a single exponentiation* over a standard 256-bit elliptic curve. Using fast curves such as Curve25519 [Ber06] or Four$\mathbb{Q}$ [CL15], we estimate a throughput of 15k to 40k OT/s on one core of a standard laptop, about four orders of magnitude faster than the best previous public-key PCF, and approaching the efficiency of the best PCF overall. Our results enable for the first time all users of a network, after a simple PKI setup, to efficiently generate arbitrary amounts of pairwise OT correlations without any interaction. The security of our construction reduces to four assumptions: the Decisional Composite Residuosity assumption, the sparse power-DDH assumption (a new —static, falsifiable, secure in the GGM— variant of DDH that we introduce), correlation-robust hash function, and a suitable weak pseudorandom function (which we instantiate using either the Goldreich-Applebaum-Raykov weak PRF [Gol00, AR16] or the Boneh-Ishai-Passelègue-Sahai-Wu weak PRF [BIP+18]).[5]

At the heart of our result is a new construction of efficient *constrained pseudorandom functions* from the Naor-Reingold PRF [NR97], for a class of constraints we term *inner-product membership* (IPM) constraints. Informally, CPRFs are pseudorandom functions where given a *constrained key* K_C for some predicate C, one can locally evaluate the PRF at all points x where $C(x) = 0$, while the output of the function still looks random on inputs x where $C(x) = 1$. A constrained key for the class IPM is associated to a vector z and a set S, and allows evaluating the PRF on an input x if and only $\langle x, z\rangle \in S$. Along the way to our efficient public-key PCF, we achieve several results of independent interest:

- We show how a simple tweak to the Naor-Reingold PRF yields a constrained PRF (CPRF) for IPM constraints, in the random oracle model. IPM constraints capture several predicates of interest, and in particular, we obtain the first puncturable pseudorandom function (CPRF for the class of point functions) in the complexity class NC^1.
- Observing that several low-complexity PRFs can be expressed as IPM constraints, we obtain a *pseudorandomly* constrained PRF: a constrained key is associated to a (low-complexity) PRF, and allows evaluating the CPRF on inputs x for which the low-complexity PRF evaluates to 0.
- We then show that pseudorandomly constrained PRFs yield *precomputable* PCFs. The notion of precomputability for PCFs was recently introduced in [CMPR23] together with a proof-of-concept (inefficient) construction from DCR. In a precomputable PCF, one of the parties can locally generate a PCF key and precompute its entire share of the correlated randomness *even before knowing the identity of the other party*. This pushes to an extreme the possibility of preparing secure computation protocols ahead of time, by allowing one party to execute the entire preprocessing before knowing its input, the function, and the identity of its opponent. Combining these results, we obtain the first *concretely efficient* precomputable PCF, from the sparse DDH assumption, in the random oracle model. While the resulting construction is not a public-key PCF, distributing the key generation of our precomputable PCF is also very simple, requiring only n parallel calls to an oblivious transfer protocol (with $n = 256$ in our most efficient instantiation).

Eventually, we explore an application of public-key PCF to *designated-verifier zero-knowledge proofs* (DV-NIZKs). A DV-NIZK allows any prover to demonstrate the truth of a statement using a single message, such that the proof can be verified using a secret verification key. DV-NIZKs are

[5] The DCR assumption in our construction can be replaced by the DDH assumption over subgroups of finite fields, but at the cost of a less efficient construction.

believed to be easier to obtain than standard NIZKs, in the following sense: they are known to exist under the plain CDH assumption in pairing-free groups [CH19, QRW19, KNYY19], while NIZKs are only known in pairing groups, or using subexponential hardness assumptions [JJ21, CJJQ23]. Yet, efficiency-wise, we do not know of any concretely efficient construction of DV-NIZKs in pairing-free groups (efficient NIZKs are known in pairing groups [GS08, KW15, CH20], and known DV-NIZKs in pairing-free groups rely on the hidden bit model, for which no concretely efficient instantiation is known). We show how, using a public-key PCF, one can compile any Σ-protocol with binary challenge into a DV-NIZK. Plugging our construction of public-key PCF, we obtain a new DV-NIZK from polynomial assumptions over pairing-free groups for all languages that admit a bit Σ-protocol, with communication comparable to that of the Σ-protocol. Conceptually, our result can be seen as observing that a public-key PCF suffices to upgrade *non-reusable* DV-NIZKs (which exist from public key encryption [CHH+07]) into *reusable* DV-NIZKs.

2 Technical Overview

In this section, we provide a detailed overview of our results. We start by describing our main paradigm about constructing PCF for OT correlations from Pseudorandomly Constrained PRFs in Section 2.1. Then, in Section 2.2, we explain how to modify the Naor-Reingold PRF in order to obtain a CPRF for the class of inner-product membership predicates, and show that various weak PRF constructions can be expressed as such predicates in Section 2.3, leading to instantiations of pseudorandomly constrained PRFs and therefore of PCFs.

Next, we focus on optimizing the resulting PCFs. In Sections 2.4 and 2.5,we provide several optimizations which benefit the most efficient instantiations of our paradigm and detail our optimized PCF construction. Concrete parameters are provided in Section 2.6.

We then describe in Section 2.7 how our PCF can be turned into an efficient public-key PCF by relying on ideas borrowed from [OSY21], and how to optimize the resulting construction.

Finally, in Section 2.8, we explain how our public-key PCF can be used to construct reusable designated-verifier NIZKs.

2.1 A PCF for OT from Pseudorandomly Constrained PRFs

Let $F = (F.\mathsf{KeyGen}, F.\mathsf{Eval})$ be a (weak, strong) PRF with key space $\mathcal{K}$ and binary outputs. For a key $K \in \mathcal{K}$, let $F_K : x \mapsto F.\mathsf{Eval}(K, x)$. Also, let $\mathsf{CPRF} = (\mathsf{CPRF.KeyGen}, \mathsf{CPRF.Eval}, \mathsf{CPRF.Constrain}, \mathsf{CPRF.CEval})$ denote a constrained PRF for the class $\mathcal{F} = \{F_K\}_{K\in\mathcal{K}} \cup \{1 - F_K\}_{K\in\mathcal{K}}$, i.e., $\mathcal{F}$ contains all predicates "$F.\mathsf{Eval}(K, x)$ evaluates to b" for $b \in \{0, 1\}$ and $K \in \mathcal{K}$. Then, we construct a (weak, strong) pseudorandom correlation function for oblivious transfer correlation as follows:

- The sender gets two independent master secret keys $(\mathsf{msk}_0, \mathsf{msk}_1)$ of the CPRF. On an input x, this party evaluates the CPRF on x using both keys to obtain two pseudorandom outputs (y_0, y_1).
- The receiver gets a random (weak) PRF key $K \xleftarrow{\$} \mathcal{K}$, and two constrained keys: ck_0 that is msk_0 constrained at "$F_K(x) = 0$", and ck_1 that is msk_1 constrained at "$F_K(x) = 1$". On an input x, this party computes $b \leftarrow F_K(x)$, and sets $y_b \leftarrow \mathsf{CPRF.CEval}(\mathsf{ck}_b, x)$. It then outputs (b, y_b).

Correctness is straightforward: for any x, the predicate $F_K(x) = b$ is satisfied for some $b \in \{0, 1\}$, hence the constrained key ck_b yields the correct output y_b by the correctness of the CPRF. *Sender security* follows from the fact that the two constrained keys are constrained at F_K and $1 - F_K$, respectively, hence both constrains can never be satisfied at the same time. Thus, by the security of the CPRF, when $F_K(x) = b$, the value y_{1-b} is indistinguishable from random for the receiver. *Receiver security* follows from the (weak) pseudorandomness of F, which entails that $b = F_K(x)$ is pseudorandom from the sender's perspective.

We sketch the full construction below (we omit the public parameters pp output by the CPRF for simplicity):

- $\mathsf{PCF.Gen}(1^\lambda)$:
  - For $b \in \{0, 1\}$, run $\mathsf{msk}_b \leftarrow \mathsf{CPRF.KeyGen}(1^\lambda)$.

  - Sample $K \leftarrow F.\mathsf{KeyGen}(1^\lambda)$, and compute $\mathsf{ck}_0 \leftarrow \mathsf{CPRF.Constrain}(\mathsf{msk}_0, F_K)$ and $\mathsf{ck}_1 \leftarrow \mathsf{CPRF.Constrain}(\mathsf{msk}_1, 1 - F_K)$.
  - Output $k_0 \leftarrow (\mathsf{msk}_0, \mathsf{msk}_1)$ and $k_1 \leftarrow (K, \mathsf{ck}_0, \mathsf{ck}_1)$.
- $\mathsf{PCF.Eval}(\sigma, k_\sigma, x)$:
  - If $\sigma = 0$, parse $k_0 = (\mathsf{msk}_0, \mathsf{msk}_1)$, and compute $y_b \leftarrow \mathsf{CPRF.Eval}(\mathsf{msk}_0, x)$ for $b \in \{0,1\}$, and output (y_0, y_1).
  - If $\sigma = 1$, parse $k_1 = (K, \mathsf{ck}_0, \mathsf{ck}_1)$, and compute $b \leftarrow F_K(x)$. Set $y_b \leftarrow \mathsf{CPRF.CEval}(\mathsf{ck}_b, x)$, and output (b, y_b).

We further observe that the resulting PCF is *precomputable* as recently defined in [CMPR23]. Informally, it allows one of the parties to locally generate its own PCF key and compute its correlated randomness entirely, before even knowing the identity of the other party. In the above construction, the sender can precompute all pairs (y_0, y_1) ahead of time, and it is therefore precomputable.

We note that the fact that CPRFs for a class containing a PRF yield a PCF is not entirely new; for example, a similar observation was briefly mentioned in [BGMM20]. However, the few known constructions of sufficiently expressive CPRFs [BV15, AMN+18, CMPR23] are too expensive, and using them within the above transformation yields PCFs that are much less flexible than generic constructions based on homomorphic secret sharing or threshold FHE (that are not restricted to the OT correlation), and much less efficient than state-of-the-art PCFs [BCG+20, BCG+22]. Our key contribution is identifying that a simple tweak to the Naor-Reingold PRF [NR97] yields an extremely efficient pseudorandomly constrained PRF as we explain below.

2.2 A CPRF for Inner-Product Membership from the Naor-Reingold PRF

Let us first recall the Naor-Reingold PRF [NR97], whose input domain is $\mathcal{X} = \{0,1\}^n$. Let $\mathbb{G} = \mathbb{G}(\lambda)$ be a family of cyclic groups of prime order $p = p(\lambda)$.

- $F.\mathsf{KeyGen}(1^\lambda)$: Sample $g \xleftarrow{\$} \mathbb{G}$ and $a_1, a_2, \cdots, a_n \xleftarrow{\$} \mathbb{Z}_p^*$. Output $\mathsf{msk} \leftarrow (g, a_1, \cdots, a_n)$.
- $F.\mathsf{Eval}(\mathsf{msk}, x)$: On input $x = (x_1, \cdots, x_n) \in \{0,1\}^n$, output $g^{\prod_{i=1}^n a_i^{x_i}}$.

Evaluating the Naor-Reingold PRF requires a few multiplications, followed by a single exponentiation. Its security reduces to the Decisional Diffie-Hellman assumption over $\mathbb{G}$.

A no-evaluation-secure CPRF for inner-product. As a warm-up, we define the class of predicates

$$C_z : x \to \begin{cases} 0 & \text{if } \langle x, z \rangle = 0 \\ 1 & \text{otherwise.} \end{cases}$$

That is, a constrained key for z allows to evaluate the PRF on all inputs x satisfying $\langle x, z \rangle = 0$. Now, consider the following extension of the Naor-Reingold PRF:

- $F.\mathsf{Constrain}(\mathsf{msk}, z)$: Sample $r \xleftarrow{\$} \mathbb{Z}_p^*$ and define $(\alpha_1, \cdots, \alpha_n) \leftarrow (r^{-z_1} \cdot a_1, \cdots, r^{-z_n} \cdot a_n)$, and output $\mathsf{ck} = (g, \alpha_1, \cdots, \alpha_n)$.
- $F.\mathsf{CEval}(\mathsf{ck}, x)$: On input $x = (x_1, \cdots, x_n) \in \{0,1\}^n$, output $g^{\prod_{i=1}^n \alpha_i^{x_i}}$.

Here, each key a_i is *blinded* by a term r^{-z_i}, and the outputs of the Eval and CEval algorithms coincide when the blinding terms cancel out which happens precisely when the inner product $\langle x, z \rangle$ is equal to 0 modulo the order of r. For a safe prime p with $p-1 = 2q$, the order of r is q or $2q$ with overwhelming probability, so with $q \gg n$, $\langle x, z \rangle = 0 \bmod q$ iff $\langle x, z \rangle = 0$ over the integers. More precisely, we have:

$$\begin{aligned} F.\mathsf{CEval}(\mathsf{ck}, x) = g^{\prod_{i=1}^n \alpha_i^{x_i}} &= g^{\prod_{i=1}^n (r^{-z_i} \cdot a_i)^{x_i}} = g^{r^{-\sum_{i=1}^n x_i z_i} \prod_{i=1}^n a_i^{x_i}} \\ &= (g^{\prod_{i=1}^n a_i^{x_i}})^{r^{-\langle x, z \rangle}} = (F.\mathsf{Eval}(\mathsf{msk}, x))^{r^{-\langle x, z \rangle}} \\ &= F.\mathsf{Eval}(\mathsf{msk}, x) \text{ iff } \langle x, z \rangle = 0 \ . \end{aligned}$$

Furthermore, when the adversary makes no query to the evaluation oracle, it can be shown that the pseudorandomness of the above construction on a challenge input x where $\langle x, z \rangle \neq 0$ holds as long as $g^{r^{\langle x, z \rangle}}$ looks random for a uniformly random $r \in \mathbb{Z}_p^*$. Indeed, the constrained key owner can compute $r^{-\langle x, z \rangle} \prod_{i=1}^n a_i^{x_i}$ and knows g, x, z. The actual evaluation is $g^{\prod_{i=1}^n a_i^{x_i}}$ and the constrained key reveals no information about r since a_i are uniformly random in $\mathbb{Z}_p^*$.

Before we move on, we make the following observations:

1. The algorithm $F.\mathsf{CEval}$ does not need to know z. Hence, our CPRF for inner products is also constraint-hiding.
2. We described the construction for an input and a constrain $x, z \in \{0,1\}^n$ for simplicity, and to match with the original construction of Naor and Reingold. However, the construction extends immediately to the setting where $x, z \in [\pm B]^n$, where B is some polynomial-size bound (the security of the original Naor-Reingold construction for general inputs of this form was shown in [ABP15] to reduce to a variant of the Diffie-Hellman assumption). We then have $|\langle x, z\rangle| \leq n \cdot B^2$ and assuming $n \cdot B^2 \ll q$, the inner product is again computed over the integers.
3. Eventually, we also consider a straightforward modification of the construction where both parties apply an arbitrary public preprocessing function $\mathsf{p}(\cdot)$ on the input x before feeding it into Eval or CEval. This allows to force the input x to have a specific format.

From no-evaluation security to full security. While the above construction can be attacked if the adversary can make an evaluation query, we recall that any no-evaluation secure CPRF can be turned into an adaptively secure CPRF (with any number of evaluation queries) in the random oracle model by hashing the output, as explained [AMN+18]. Hence, proving no-evaluation security is sufficient for our purpose.

In fact, an even simpler construction exists in the random oracle model by simply removing the group and considering only the exponents. Precisely, define the PRF output as $H(\prod_{i=1}^{n} a_i^{x_i})$ where H is a hash function modeled as a random oracle. Next, define the constrained key as $(\alpha_1, \ldots, \alpha_n)$, as in the construction above. For this CPRF, security holds unconditionally in the ROM, since the evaluation is uniform unless the adversary can compute $r^{\langle x,z\rangle}$, which is hard since the adversary has no information about r and it has high min-entropy (and its order is at least $q \gg n$).

While we do hash the output to obtain full security, we still rely on the above construction, instantiated with a group $\mathbb{G}$, as it comes with benefits. Indeed, we can add elements of the form g^{r^t} in the constrained key which turns out to be extremely useful, as we describe below.

From inner product to inner product membership. Previously, we showed how the Naor-Reingold PRF can be turned into a CPRF for inner-product predicates. Yet, this class of predicates is too restricted for instantiating the PCF construction explained in Section 2.1. Our next observation is that this class can be significantly expanded by adding elements of the form g^{r^t} to the constrained key to help the evaluator cancel out some r^{-t} terms. Indeed, if the evaluator uses g^{r^t} instead of g as the basis for exponentiation, the computation of $F.\mathsf{CEval}(\mathsf{ck}, x)$ becomes

$$\begin{aligned} F.\mathsf{CEval}(\mathsf{ck}, x) &= (g^{r^t})^{\prod_{i=1}^{n}(r^{-z_i}\cdot a_i)^{x_i}} \\ &= (g^{r^{t-\langle x,z\rangle}})^{\prod_{i=1}^{n} a_i^{x_i}} \\ &= (F.\mathsf{Eval}(\mathsf{msk}, x))^{r^{t-\langle x,z\rangle}} \quad , \end{aligned}$$

which is the same as $F.\mathsf{Eval}(\mathsf{msk}, x)$ if and only if $t = \langle x, z\rangle$. What makes this observation particularly powerful is that the evaluator can be given terms of the form g^{r^t} *for multiple values of* t, and choose upon evaluation the term g^{r^t} where $t = \langle x, z\rangle$. This yields a CPRF for the class of predicates

$$C_{z,S} : x \to \begin{cases} 0 & \text{if } \langle x, z\rangle \in S \\ 1 & \text{otherwise} \end{cases}$$

where $S \subseteq \mathbb{Z}_{p-1}$ is a polynomial size subset. The full construction is given below:

- $F.\mathsf{Constrain}(\mathsf{msk}, z, S)$: Sample $r \xleftarrow{\$} \mathbb{Z}_p^*$. For every $t \in S$, define $g_t \leftarrow g^{r^t}$, and let $(\alpha_1, \cdots, \alpha_n) \leftarrow (r^{-z_1} \cdot a_1, \cdots, r^{-z_n} \cdot a_n)$. Output $\mathsf{ck} = ((g_t)_{t\in S}, \alpha_1, \cdots, \alpha_n)$.
- $F.\mathsf{CEval}(\mathsf{ck}, x)$: On input $x = (x_1, \cdots, x_n) \in [\pm B]^n$, let $t \leftarrow \langle x, z\rangle$. Output $g_t^{\prod_{i=1}^{n} \alpha_i^{x_i}}$.

Note that our prior claim that the constrained key contains no information about r does not longer hold as it now contains the extra elements g^{r^t} for all $t \in S$, hence no-evaluation security is no longer unconditional. We show that this construction is (no-evaluation) secure under a variant of the Diffie-Hellman assumption which we call *sparse power-DDH* assumption. The sparse power-DDH assumption states that for a subset $S \subseteq [\ell]$, where $\ell \in \mathbb{N}$ is polynomially-bounded, given g^{r^t} for various $t \in S$, it is infeasible to distinguish g^{r^t} for $t \in [\ell] \setminus S$ from uniformly random group elements.

The sparse power-DDH assumption is a static falsifiable assumption (which holds unconditionally in the generic group model), and it can be viewed as a natural generalization of the power-DDH assumption (which states that given g^{r^i} for $i \in \{1, \ldots, n\}$, it is infeasible to distinguish $g^{r^{n+1}}$ from random).

On IPM predicates. The inner product membership predicate captures several predicates of interest. We already mentioned inner-product equality (in which case our CPRF is constraint-hiding), and it also captures inner-product *in*equality. Moreover, this class captures puncturing, which, to our knowledge, yields the first candidate puncturable pseudorandom function in the complexity class NC^1 (assuming that the hash function is instantiated with an NC^1 function).[6] Since puncturable PRFs have independent applications, for instance in the context of indistinguishability obfuscation (which is typically first built for NC^1 before being bootstrapped to P), we expect that this result could have other applications. Furthermore, the IPM predicates capture several variants of puncturing, such as puncturing a Hamming ball, and many more. Most importantly, and as we explain in the following section, this class of predicates captures several candidate weak pseudorandom functions from the literature, which makes it powerful enough to instantiate the PCF for OT correlation outlined in Section 2.1.

2.3 Inner-Product Membership Weak Pseudorandom Functions

It turns out that several candidate weak PRFs from the literature can be expressed as IPM predicates. In this section, we provide a non-exhaustive list of such constructions. Notably, we show that the BIPSW [BIP+18] as well as the XOR-MAJ [Gol00, AL16] candidate weak PRFs fall into this category and lead to efficient instantiations of our paradigm. We write IPM-wPRF to denote a weak PRF expressed as an IPM predicate.

The Learning-with-Rounding wPRF. Given a modulus q and a smaller modulus $q' \ll q$, the well-known candidate wPRF of [BPR12] is given by $F_z(x) = \lfloor \langle x, z \rangle \rceil_{q'}$, where $x, z \in \mathbb{Z}_q^n$, and $\lfloor \cdot \rceil_{q'}$ denotes an appropriate procedure for rounding to an element of $\mathbb{Z}_{q'}$. This candidate was shown in [BPR12] to be a secure wPRF under the standard LWE assumption, provided that q, q' are superpolynomial. This proof was further refined in [AKPW13] to show that a polynomial-size modulus suffices for the reduction. While the proof does not extend to the case $q' = 2$, no known efficient attack is known, and the learning-with-rounding (LWR) assumption is now widely conjectured to hold even outside of the regime where it reduces to LWE. When q is polynomial and $q' = 2$, we observe that it suffices to define S as $S = \{s \in \mathbb{Z}_{n \cdot q^2} \ : \ \lfloor (s \mod q) \rceil_2 = 0\}$ to rewrite this wPRF as an IPM-wPRF (with $|S| \approx n \cdot q^2/2$). Due to the q^2 overhead in the size of S, for standard choices of the modulus q, our construction with this candidate does not yield a very efficient instantiation. However, it forms a basis for our next candidate.

The BIPSW wPRF. In [BIP+18], the authors introduced several new low-complexity wPRF candidates, together with some preliminary analysis to back up the security claims. Five years later, these candidates have received some attention, both by cryptanalysts [CCKK21, JMN23] and in the context of a range of applications, from secure computation to side-channel security [DGH+21, ADDG23, DMMS21]. As the authors observed, one of their candidates (that we denote as BIPSW) can be rephrased as an LWR-style wPRF: $F_z(x) = \lfloor \langle x, z \rangle \bmod 6 \rceil$, with $x, z \in \{0,1\}^n$, and with the rounding function defined as $\lfloor s \rceil = 0$ if $(s \bmod 6) \in \{0, 1, 2\}$, and $\lfloor s \rceil = 1$ if $(s \bmod 6) \in \{3, 4, 5\}$. The authors initially suggested a key length $n = 384$ as a conservative choice for security. Several attacks were later shown, in [CCKK21] and very recently in [JMN23], suggesting that the key length should be increased to $n = 770$. We note that the BIPSW candidate fits particularly well in our framework: it can be written as an IPM-wPRF by defining $S = \{s \leq n \ : \ \exists k \leq n/6, i \in \{0,1,2\}, s = 6k + i\}$. The size of S is $n/2$, which is as low as $|S| = 385$ for $n = 770$.

The Goldreich-Applebaum-Raykov wPRF. In [Gol00], Goldreich suggested an approach for building one-way functions by evaluating a fixed low-arity predicate on fixed random small subsets of the input bits (*i.e.*, $f(x) = (P(x[S_1]), \cdots, P(x[S_m])$, where P is a predicate and $S_1, \cdots, S_n$ are

[6] It is not too hard to build a PPRF in NC^1 by following the blueprint of the GGM PRF [GGM84b] but using a λ-ary tree instead of a binary tree and instantiating the PRG with an NC^0 PRG with polynomial stretch. However, such constructions are inherently limited to superpolynomial-size domains, while our construction can handle subexponential-size domains.

fixed random subsets, also $x[S_i]$ denotes the substring of the bits of x indexed by S_i). Later works suggested that for a suitable choice of P, these *random local functions* can also be conjectured to be a pseudorandom generator when $m > |x|$. The construction of Goldreich has ever since been featured extensively in cryptography, both by cryptanalysts [CM01, MST03, BQ09, OW14, CEMT14, ABR16, AL16, LV17, CDM+18, AK19, OST19, Méa, YGJL21, Méa22, Üna23b, DMR23, Üna23a] and in numerous cryptographic applications such as low-complexity cryptography, secure computation, obfuscation, and many more [App12, AR16, BCG+17, App17, JLS21] (see [App15] for a survey from 2015). In [AR16], Applebaum and Raykov showed how Goldreich's random local functions can also yield plausible candidate wPRFs for suitable choices of P, when $|S_i| = \Omega(\log n)$. We denote this candidate wPRF as Goldreich-Applebaum-Raykov(GAR) wPRF.

We outline how the GAR wPRF can be expressed as an IPM-wPRF. The core idea is to view the random input x as an *encoding* of a subset $S_x \subset [n]$ of size $|S_x| = \Omega(\log n)$, and to preprocess x using some fixed preprocessing function g such that $\langle g(x), z\rangle = z[S_x]$. To do so, we let $|x| = \log^2(n)$ and parse x as a $k = \Omega(\log(n))$-tuple of distinct indices $j_1, \cdots, j_k \in [n]$ by keeping only the distinct j's from the $\log(n)$ strings $j \in \{0,1\}^{\log(n)}$ (viewed as elements of $[n]$). Then, we let $g(x)$ be the length-n vector defined as follows: the vector $g(x)$ is 0 everywhere, except that it has the entry $2^{\ell-1}$ at position j_ℓ for $\ell = 1, \ldots, k$. Observe that with this encoding, computing $\langle g(x), z\rangle$ returns $\sum_{\ell=1}^{k} z_{j_\ell} \cdot 2^{\ell-1}$, which is exactly the integer whose binary representation encodes the subset S_x of the bits of z. Now, for any choice of $\log(n)$-ary predicate P, we define $S = \{s \in [2^k] \ : \ P(s) = 0\}$. Observe that checking whether $\langle g(x), z\rangle \in S$ is equivalent to computing $P(z[S_x])$, where S_x is the $\Omega(\log n)$-sized subset defined by x. Hence, up to the preprocessing of the input x (which is for free in our construction), the GAR wPRF can be expressed as an IPM-wPRF. As long as the arity of P is $k = O(\log(n))$, we have $|S| = O(2^k) = \mathsf{poly}(n)$.

This yields a generic construction that works for any choice of predicate with sufficiently low arity. Directly instantiating this construction does not yield a very competitive PCF. However, our next observation is that for the most standard and well-studied choices of predicate P, the generic construction can be considerably improved.

The XOR-MAJ wPRF. The previous construction works for arbitrary predicates P, provided that P takes at most $O(\log n)$ bits as input. In this section, we observe that when P is of the form $P(x_0, x_1) = \mathsf{SYM}_0(x_0) \oplus \mathsf{SYM}_1(x_1)$, where $\mathsf{SYM}_0, \mathsf{SYM}_1$ are arbitrary symmetric functions, then there exists an improved construction that handles predicates of *arbitrary* locality. This capture in particular the XOR-MAJ predicate, which computes the XOR between the parity of the x_0 input and the majority of the x_1 input. XOR-MAJ is probably the most common choice of predicate for the GAR wPRF, and its properties have been studied extensively [AL16, CDM+18, Méa, YGJL21, Méa22, Üna23b, DMR23].

We briefly outline how to express the GAR wPRF with the XOR-MAJ predicate as an IPM-wPRF (the generalization to other symmetric functions is immediate). Assume that the predicate is $P = \mathsf{XOR_k\text{-}MAJ}_\ell$, which takes as input a $(k+\ell)$-bit subset z of the bits of the secret key, and outputs $\mathsf{XOR}(z_1, \cdots, z_k) \oplus \mathsf{MAJ}(z_{k+1}, \cdots, z_{k+\ell})$. Similarly to before, we parse a random input x as an encoding of two random disjoint subsets $(S_{0,x}, S_{1,x})$ of $[n]$, of size k and ℓ respectively. Then, we let $\mathsf{p}(x)$ denote the length-n vector with 1's at all entries indexed by $S_{1,x}$, value $\ell + 1$ at all entries indexed by $S_{0,x}$, and 0's everywhere else. Observe that this encodings yields

$$\langle \mathsf{p}(x), z\rangle = \mathsf{HW}((z_i)_{i \in S_{1,x}}) + (\ell+1) \cdot \mathsf{HW}((z_i)_{i \in S_{0,x}}),$$

where $\mathsf{HW}(\cdot)$ denotes the Hamming weight. Furthermore, since $|S_{1,x}| = \ell$, every integer $\langle \mathsf{p}(x), z\rangle$ computed as above uniquely determines the pair $(\mathsf{HW}((z_i)_{i \in S_{1,x}}), \mathsf{HW}((z_i)_{i \in S_{0,x}})))$. In turn, symmetric functions such as XOR and MAJ are uniquely determined by the Hamming weight of their inputs (in particular, $\mathsf{XOR}(z)$ returns $\mathsf{HW}(z)$ mod 2 and $\mathsf{MAJ}(z)$ returns 1 iff $\mathsf{HW}(z) > \ell/2$). Then, using the fact that $A \oplus B = 0$ iff $A = B$, we define S as follows:

$$S = \{s = s_1 + (\ell+1)s_0 \in [\ell + (\ell+1) \cdot k] \ : \ [\mathsf{HW}(s_1) \bmod 2] = [\mathsf{HW}(s_0) > \ell/2]\}.$$

Compared to the previous construction, this new construction is tailored to XOR-MAJ (or more generally to predicates of the form $P(x_0, x_1) = \mathsf{SYM}_0(x_0) \oplus \mathsf{SYM}_1(x_1)$[7].). However, for a predicate of

[7] even more generally, the construction can be adapted to handle the XOR of any number N of symmetric predicates with respective locality $\ell_1, \cdots, \ell_N$, with $|S| = O(\prod_{i=1}^{N} \ell_i)$

locality $\ell + k$, the size of S scales as $O(\ell \cdot k)$, which is an exponential improvement over the $2^{\ell+k}$ cost of the generic construction. While the GAR wPRF is typically considered in the low-locality setting, our construction allows simultaneously relying on a particularly conservative parameter setting, using XOR-MAJ with locality $O(\sqrt{n})$ (in this parameter regime, the GAR wPRF is generally conjectured to provide subexponential security $2^{O(\sqrt{n})}$), and keeping S to a small size $|S| = O(n)$. Together with the BIPSW wPRF candidate, this instantiation yields the most efficient concrete instantiations of our framework. To give a single data point, using the state-of-the-art cryptanalysis on Goldreich-style local wPRFs, we can set the key length n to 256 and use the XOR_{10}-MAJ_{64} predicate to achieve 128 bits of security for up to 2^{40} queries to the wPRF, and have $|S| = 357$.

Other candidates. Before moving on, we note that several other candidate wPRFs can be shown to fit in the IPM-wPRF framework: candidates based on sparse LPN or variable-density LPN [BCG+20], and the BFLK candidate $f_{A,B} : x \mapsto (\bigoplus_{i\in A} x_i) \oplus \mathsf{MAJ}((x_i)_{i\in B})$ from [BFKL94]. These candidates yield less competitive instantiations of our framework, and we do not discuss them further here. However, this suggests two interesting open questions for future work:

- Finding the best-possible IPM-wPRF (*i.e.* the one achieving the best possible tradeoffs between security and key size + size of S), and
- All candidates considered in this work are *weak* PRFs. Whether there exists *strong* IPM-PRFs remains an interesting open question (though we note that any IPM-wPRF can be generically upgraded to a strong PRF when instantiating the preprocessing function p with a random oracle).

2.4 Optimizations

The above framework can be largely improve by various optimizations. In this section, we sketch several optimizations that allow improving the performance of our PCF.

Halving the key size. When we instantiate the framework of Section 2.1 using our Naor-Reingold CPRF, the sender key consists of two master secret keys $(g, a_1, \cdots, a_n)$ and $(h, a'_1, \cdots, a'_n)$, and the receiver key for the predicate $F_{z,S} : x \mapsto \langle x, z\rangle \in_? S$ consists of $(r^{-z_i} \cdot a_i)_{i\in[n]}, (r'^{-z_i} \cdot a'_i)_{i\in[n]}, (g^{r^t})_{t\in S}$, and $(h^{r'^t})_{t\in[\pm n\cdot B^2]\setminus S}$ (where r, r' are random elements of $\mathbb{Z}_p^*$ and B is a bound on the entries of x, z). Thanks to the random self-reducibility of DDH, the two master secret keys can actually use the same elements $(a_1, \cdots, a_n)$ provided that they use different bases g, h. For the same reason, we can also set $r = r'$ without any security loss. This reduces the sender key size by a factor two, and significantly compresses the receiver key size as well. Concretely, we can use the following keys:

- Sender key: $(g, h, a_1, \cdots, a_n)$,
- Receiver key: $(r^{-z_i} \cdot a_i)_{i\le n}, (g_t)_{t\in[\pm n\cdot B^2]}$,

where $g_t \leftarrow g^{r^t}$, for $t \in S$, and $g_t \leftarrow h^{r^t}$, for $t \in [\pm n \cdot B^2] \setminus S$. The resulting construction can be proven secure under the same assumptions as the basic construction.

Reusing the g_t's. We observe that the value z (which depends on the underlying PRF key used by the receiver) is only known to the receiver, while the set S is public and is related to the definition of the PRF. In a multiparty setting where the sender wants to compute PCF keys with multiple receivers, we can exploit this observation to define the g_t's once for all, and pass them as common parameters to be used by all receivers. This requires adding two additional terms $(a_{0,j}, a'_{0,j})$ in the sender key for each receiver R_j, to re-randomize the bases g, h. That is, the sender now computes its pseudorandom OT messages as

$$y_{0,j}^{(x)} \leftarrow g^{a_{0,j}\cdot\prod_{i=1}^n a_i^{x_i}} \qquad\qquad y_{1,j}^{(x)} \leftarrow h^{a'_{0,j}\cdot\prod_{i=1}^n (a_i)^{x_i}},$$

where $g^{a_{0,j}}$ and $h^{a'_{0,j}}$ will play the role of fresh new bases for each receiver R_j (the receiver CEval have to be adapted accordingly). With this modification, the g_t's can be thought of as public parameters or rather as a kind of public key associated to the sender.

Compressing the a_i's. When instantiating the group with a suitable elliptic curve, the size of the a_i's is typically 2λ bits (to achieve λ bits of security against generic discrete log attacks). To further reduce the key size, the a_i's can be generated from a pseudorandom generator in a two-step fashion: first, the sender receives a λ-bit seed seed and computes $(g, h, \mathsf{seed}_1, \cdots, \mathsf{seed}_n) \leftarrow \mathsf{PRG}(\mathsf{seed})$, where

$\mathsf{PRG} : \{0,1\}^\lambda \mapsto \{0,1\}^{(4+n)\cdot\lambda}$ (each seed_i is in $\{0,1\}^\lambda$). Second, define $a_i \leftarrow \mathsf{PRG}'(\mathsf{seed}_i)$, where $\mathsf{PRG}' : \{0,1\}^\lambda \mapsto \{0,1\}^{2\lambda}$. The advantage of this approach is that it enables compressing both the sender key and the receiver key:

- The sender key is now simply the λ-bit seed seed.
- The receiver key is still $(r^{-z_i} \cdot a_i)_{i \le n}$ (together with the public g_t terms), except for the indices i where $z_i = 0$, where it holds that $r^{-z_i} \cdot a_i = a_i$. For such indices, we can send simply send seed_i, which is twice smaller. When z_i is a bit-string (which is the case for the BIPSW and XOR-MAJ wPRFs), this reduces the size of about half of the $r^{z_i} \cdot a_i$ to that of seed_i, resulting in a 25% reduction of the key length.

Exploiting the structure in S. Assume that S contains all integers s (from some bounded range $\{0, \cdots, m \cdot R\}$) such that $(s \bmod m) < m/2$, where m is some fixed value; we say that S is m-antiperiodic. Then we *almost* get the following equivalence:

$$s \notin S \iff (s - m) \in S \ ,$$

where the *almost* stems from the fact that the equivalence breaks down at the extremities: for example, $s = m/2 + 1 \notin S$, yet $s - m \notin S$ because $s - m$ is outside of the bounded range $\{0, \cdots, m \cdot R\}$. Nevertheless, we can recover the equivalence by slightly extending S into $S' = S \cup \{-m/2, \cdots, -1\}$ (the equivalence becomes: for every $s \in \{0, \cdots, R \cdot m\}$, $s \notin S' \iff (s - m) \in S'$).

In this case, we observe that it is not necessary to include in the receiver key both $(g_t)_{t \in S}$ and $(g_t)_{t \notin S}$. Indeed, as we are constraining before a key msk_0 with respect to the predicate "$\langle x, z\rangle \in S$", and a second key msk_1 with respect to the predicate "$\langle x, z\rangle \notin S$", we can then rewrite the second constraint as "$\langle x, z\rangle - m \in S$". Concretely, we now deal a single key msk to the sender, but we add an $(n+1)$-th element a_{n+1} to act as a *shift*. The keys become:

- Sender key: $\mathsf{msk} = (g, a_1, \cdots, a_n, a_{n+1})$
- Receiver key: $\mathsf{ck} = (g, r^{-z_1} \cdot a_1, \cdots, r^{-z_n} \cdot a_n, r^m \cdot a_{n+1}), (g_t)_{t \in S'}$.

Given msk, on input x the sender computes their two OT inputs as $y_b \leftarrow \mathsf{CPRF.Eval}(\mathsf{msk}, x|b)$ for $b = 0, 1$. That is, we have:

$$y_b \leftarrow g^{\prod_{i=1}^n a_i^{x_i} \cdot a_{n+1}^b} \text{ for } b = 0, 1.$$

Now, thanks to the term $r^{-m} \cdot a_{n+1}$ in the receiver key, for every input x, there is only a single $b \in \{0,1\}$ such that $\langle x|b, z| - m\rangle \in S$, i.e. such that $\langle x, z\rangle - b \cdot m \in S'$. Compared to the previous construction, this (almost) halves the number of group elements g_t in the receiver key, going from $m \cdot R$ to $(m/2) \cdot (R+1)$.

The BIPSW wPRF and the XOR-MAJ wPRF satisfy this property: the set S is m-antiperiodic with $m = 6$ for BIPSW, and $m = 49$ for our parameter choice with XOR-MAJ. Hence, this optimization can also be applied to these two instantiations.

2.5 Final PCF Construction

We are now fully equipped to describe our final PCF construction. Concrete parameters for both instantiations based on the BIPSW and the XOR-MAJ are provided in the next section. Let the input domain be $\{0,1\}^\ell$. Let F be an $\mathsf{IPM\text{-}wPRF}$ with preprocessing function $\mathsf{p} : \{0,1\}^\ell \mapsto [0, B]^n$ (for some polynomial bound B), key space $\{0,1\}^n$ and associated set S; that is, given a key $z \xleftarrow{\$} \{0,1\}^n$ and an input $x \in \{0,1\}^\ell$, $F_z(x)$ outputs 1 iff $\langle \mathsf{p}(x), z\rangle \in S$. We assume that S is m-antiperiodic for some integer m (*i.e.* $S = \{s \in \{0, \cdots, R\} \ : \ s \bmod m < m/2\}$ for some polynomial bound R). Define $S' \leftarrow S \cup \{-m/2, \cdots, -1\}$. Fix a family of cyclic groups $\mathbb{G} = \mathbb{G}(\lambda)$ of order $p = p(\lambda)$. Let $G_0 : \{0,1\}^\lambda \mapsto \mathbb{Z}_p^* \times \{0,1\}^{n\cdot\lambda}$, $G_1 : \{0,1\}^\lambda \mapsto \mathbb{Z}_p^*$, and $G_2 : \{0,1\}^\lambda \mapsto \{0,1\}^n$ be three pseudorandom generators. Let $H : \mathbb{G} \mapsto \{0,1\}^\lambda$ be a hash function.

- $\mathsf{PCF.Gen}(1^\lambda)$: sample $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$. let $(g, \mathsf{seed}_1, \cdots, \mathsf{seed}_{n+1}) \leftarrow G_0(\mathsf{seed})$ and $a_i \leftarrow \mathbb{G}_1(\mathsf{seed}_i)$ for $i = 1$ to $n+1$. Sample $\mathsf{seed}_z \in \{0,1\}^\lambda$, $r \xleftarrow{\$} \mathbb{Z}_p^*$, and let $z \leftarrow G_2(\mathsf{seed}_z)$. For $i = 1$ to n, set $v_i \leftarrow \mathsf{seed}_i$ if $z_i = 0$, and $v_i \leftarrow r^{-1} \cdot a_i$, otherwise. Set $v_{n+1} \xleftarrow{\$} r^m \cdot a_{n+1}$. Define $g_t \leftarrow g^{r^t}$ for every $t \in S'$. Output $k_0 \leftarrow \mathsf{seed}$ and $k_1 \leftarrow (\mathsf{seed}_z, v_1, \cdots, v_{n+1}, (g_t)_{t \in S'})$.

– $\mathsf{PCF.Eval}(0, k_0, x)$: recompute $(g, \mathsf{seed}_1, \cdots, \mathsf{seed}_{n+1}) \leftarrow G_0(k_0)$ and $a_i \leftarrow \mathbb{G}_1(\mathsf{seed}_i)$ for $i = 1$ to $n+1$. Let $(s_1, \cdots, s_n) \leftarrow \mathsf{p}(x)$. Define

$$y_b \leftarrow H\left(g^{\prod_{i=1}^{n} a_i^{s_i} \cdot a_{n+1}^{b}}\right) \text{ for } b = 0, 1.$$

Output (y_0, y_1).

– $\mathsf{PCF.Eval}(1, k_1, x)$: parse k_1 as $(\mathsf{seed}_z, v_1, \cdots, v_{n+1}, (g_t)_{t \in S'})$. Let $(s_1, \cdots, s_n) \leftarrow \mathsf{p}(x)$. Recompute $z \leftarrow G_2(\mathsf{seed}_z)$. For $i = 1, \ldots, n+1$, set $\alpha_i \leftarrow G_1(v_i)$ if $z_i = 0$ or $i = n+1$, and $\alpha_i \leftarrow v_i$ else. Let $b \leftarrow F_z(x)$ and $t \leftarrow \langle \mathsf{p}(x), z \rangle - b \cdot m$. Note that by definition, this means that $t \in S'$. Define

$$y_b \leftarrow H\left(g_t^{\prod_{i=1}^{n} \alpha_i^{s_i} \cdot \alpha_{n+1}^{b}}\right).$$

Output (b, y_b).

To state our main theorem, we define the sparse power-DDH assumption with respect to S'. For a group $\mathbb{G}_\lambda = \langle g \rangle$ of prime order p, the sparse power-DDH assumption with respect to S' over a support $[0, R]$ states that

$$\left(g, (g^{r \cdot a^i})_{i \in S'}, (g^{r \cdot a^i})_{i \in [0,R] \setminus S'}\right) \overset{\mathrm{c}}{\approx} \left(g, (g^{r \cdot a^i})_{i \in S'}, (g^{t_i})_{i \in [0,R] \setminus S'}\right),$$

where $\overset{\mathrm{c}}{\approx}$ denotes computational indistinguishability, $a, r \xleftarrow{\$} \mathbb{Z}_p^*$, and $t_i \xleftarrow{\$} \mathbb{Z}_p^*$ for all $i \in [0, R] \setminus S'$. Note that the bound R and the set S' are fixed parameters of the construction; hence, this assumption is a static, falsifiable variant of the power-DDH assumption used in several previous works (e.g. [AMN+18]). It can be shown to hold in the generic group model. We obtain the following theorem:

Theorem 1 (informal). *Assuming that the sparse power-DDH assumption with respect to S' holds, that (G_0, G_1, G_2) are pseudorandom generators, that F is a secure* IPM-wPRF*, and modeling H as a random oracle, then the above construction is a weak pseudorandom correlation function for the oblivious transfer correlation.*

Note that, in the random oracle model, the construction can be upgraded to a strong PCF by first hashing the inputs [BCG+20] and can also be proven secure under a weaker *search* version of the sparse power-DH assumption.

Distributed key generation. A useful feature of our PCF is that it admits a very efficient two-round distributed key generation algorithm. Concretely, and borrowing the notations from the construction above, the OT sender can simply generate seed and r themself, and send $(g_t)_{t \in S'}$ to the OT receiver directly, together with $v_{n+1} = r^m \cdot a_i$. Then, the OT receiver samples seed_z. Eventually, to obtain the missing v_i's, observe that $v_i = \mathsf{seed}_i$ if $z_i = 0$, and $v_i = r^{-1} \cdot a_i$ otherwise. Therefore, the sender and the receiver simply run n parallel instances of an oblivious transfer protocol, where the sender input pairs $(\mathsf{seed}_i, r \cdot a_i)$, and the receiver uses selection bits z_i. Security follows immediately from the security of the oblivious transfer protocol. Using a two-round OT protocol, the entire distributed key generation can be done in two rounds, and the communication boils down to n parallel OTs plus sending $|S'|$ group elements.

2.6 Concrete Parameters

With all the above optimizations in mind, we provide two concrete instantiations of PCF for the OT correlation, using either the BIPSW wPRF candidate, or the XOR-MAJ wPRF candidate.

Curve and exponentiations. To estimate the runtime of our constructions, we rely mainly on the website `zka.lc`, which provides an extensive list of benchmarks for standard operations on various curves and over various platforms. According to the benchmarks of `zka.lc`, computing one exponentiation represents about 50μs of computation on one core of an AWS platform using curve25519 [Ber06]. Note that in our construction, the sender must compute two exponentiations (to compute (y_0, y_1)) while the receiver computes a single exponentiation. However, the two sender exponentiations use a fixed basis g. Hence, the exponentiations can be significantly sped up with

precomputation (in contrast, the receiver does an exponentiation with a basis g_t which is chosen based on the input). In our instantiations, exponentiations will generally dominate the runtime. Using more efficient curves, such as Microsoft's Four$\mathbb{Q}$ curve [CL15], the exponentiation time can be reduced to about 15μs on a Haswell architecture (note that the curve offers slightly less security compared to curve25519, about 122 bits instead of 128).

Parameters with BIPSW. For BIPSW, we used the state-of-the-art cryptanalysis from the works of [CCKK21, JMN23], and set the key length to $n = 770$, which achieves 128 bits of security according to these attacks. We note that this parameter choice ignores some significant polynomial factors in the cost estimation (that come from a nearest neighbor search), hence our parameter choice takes a bit of margin. Furthermore, the recent attack of [JMN23] has a much higher memory requirement compared to previous attack. On the other hand, we warn the reader that the BIPSW candidate is a relatively young wPRF and while a total break would be surprising at this point, the state of cryptanalysis is likely to improve over the years. With $n = 770$, S is 6-antiperiodic and we have $|S'| = 388$ With this parameter choice, the precomputable PCF has the following efficiency features:

- Key size: the receiver key size is 30.2kB (and the sender key size is 16 Bytes). Out of that, 12.1kB are public parameters $(g_t)_{t\in S'}$, which the sender can reuse with other receivers.
- Computation: computing s_b involves 385 multiplications over $\mathbb{Z}_p^*$, one exponentiation, and one hash. This translates to about 10k OT/s per core using curve25519 on an AWS platform, or about 15k OT/s per core using a curve such as Four$\mathbb{Q}$ on a Haswell architecture.

Parameters with XOR-MAJ. For the GAR wPRF instantiated with the XOR-MAJ predicate, we rely on the state-of-the-art cryptanalysis results from [AL16, CDM+18, YGJL21, Ünа23a]. Specifically, according to Table 1 of [Ünа23a], for a candidate to achieve λ bits of security with a key of length $n = \lambda^\delta$ and a bound n^{1+e} on the number of queries, the underlying predicate P must have

- rational degree at least $\frac{\delta}{\delta-1} \cdot e + 1$, and
- resiliency at least $2e + 1$.

A k-variable Boolean function P has rational degree d if it is the smallest integer for which there exist degree d polynomials g and h, not both zero, such that $P \cdot g = h$. [8] A k-variate boolean function is t-resilient if it has no nontrivial correlation with any linear combination of at most t of its inputs. We note that Table 1 of [Ünа23a] ignores the guess-and-decode attack of [YGJL21], because their attack does not have a closed-form formula. However, Both the guess-and-determine attack of [CDM+18] and the guess-and-decode attack of [YGJL21] are specifically targeted at predicates with a very small locality (the papers consider localities from 5 to 8), and their complexity scales very poorly for predicates with a larger locality. As we will see shortly, our candidates have considerably higher locality (e.g. 74 in our main instantiation) and after selecting them, we verified individually that they yield concrete instances which are (way) out of reach of the guess-and-determine and the guess-and-decode attacks. In the following, we therefore use the two criteria above to select our candidates.

The algebraic immunity and resiliency of the XOR-MAJ predicate have been studied in several papers. To match the above two constraints, it suffices to use the XOR_{ℓ_1}-MAJ_{ℓ_2} predicate with $\ell_1 = 2 \cdot (e + 1)$ and $\ell_2 = 2\delta e/(\delta - 1)$. We outline below a concrete choice of parameters for illustration: set $\delta = 1.143$. This yields $\delta/(\delta - 1) = 8$ and $n = \lambda^\delta = 256$ using $\lambda = 128$. We get $\ell_2 = 16e$, and $|S'| = 16e^2 + 33e + 2$. Setting $e = 4$, the parties can generate up to $n^{1+e} = 2^{40}$ pseudorandom OTs and $|S'| = 390$. With this parameter choice, the precomputable PCF has the following efficiency features:

- Key size: the receiver key size is 18kB (and the sender key size is 16 Bytes). Out of that, 12.2kB are public parameters $(g_t)_{t\in S'}$, which the sender can reuse with other receivers.
- Computation: computing s_b involves 74 multiplications over $\mathbb{Z}_p^*$, one exponentiation, and one hash. This translates to about 15k OT/s per core using curve25519 on an AWS platform, or about 40k OT/s per core using a curve such as Four$\mathbb{Q}$ on a Haswell architecture.

Note that other choices of parameters can yield different trade-offs, such as achieving slightly smaller key size, slightly more OTs, or slightly less computation. For example, using $\delta = 1.2858$ yields $n = 512$, $|S'| = 222$, a slightly larger key size 18.9kB, 46 multiplications instead of 74, and a bound of 2^{45} on the target number of OTs.

[8] Table 1 of [Ünа23a] mentions only the degree of the predicate, but strengthening the requirement to the rational degree is known to be necessary [AL16, DMR23].

2.7 Public Key PCF

We finally describe a *public key PCF*. Informally, a public key PCF allows users to generate a pair of public/secret keys, and then to broadcast their public key using a single message, while storing their secret key locally. Then, any pair of users can *non-interactively* obtain a PCF key pair (k_0, k_1) by combining their secret key and the other party's public key.

While the distributed key generation protocol described in Section 2.5 is particularly efficient, it requires two rounds of interaction. The protocol we now describe uses a *single* round of interaction. A major advantage of such protocol is that they enable n parties over a network to execute $\Omega(n^2)$ pairwise PCF key generations (to set up an OT channel between each pair of parties) using only $O(n)$ communication in total (this is similar to how non-interactive key exchange enable n^2 pairs of parties to agree on shared keys using $O(n)$ communication).

A simple construction. In our interactive protocol, sending $(g_t)_{t\in S'}$ does not require interaction: the interaction stems entirely from the OTs. We start with a protocol that replaces the two-round OT with the non-interactive OT protocol of Bellare and Micali [BM90]. The objective is, for the sender with input r, and the receiver with input z_i, to distributively generate keys $a_i \in \mathbb{Z}_p^*$ and $\alpha_i = r^{-z_i} \cdot a_i \in \mathbb{Z}_p^*$ respectively. These values can be viewed as multiplicative shares over $\mathbb{Z}_p^*$ of r^{-z_i} (up to inverting a_i locally). We observe that if DDH holds over (a suitable subgroup of) $\mathbb{Z}_p^*$, such multiplicative shares can be directly obtained via the Bellare-Micali protocol. Concretely, let $\mathbb{G}'$ be a suitable cyclic subgroup of $\mathbb{Z}_p^*$ where DDH is conjectured to hold, and let (G, H) be two random generators of $\mathbb{G}'$. Assume that $r \in \mathbb{G}'$. The protocol simply consists in having the sender send an ElGamal encryption of r, while the receiver sends a Pedersen commitment to z_i:

- **Sender to receiver:** pick a random coin ρ, and sends the ElGamal ciphertext $(C_0, C_1) \leftarrow (G^\rho, H^\rho \cdot r)$.
- **Receiver to sender:** pick n random coins $(\theta_1, \cdots, \theta_n)$ and send the Pedersen commitments $(H_1, \cdots, H_n) \leftarrow (H^{-z_i} \cdot G^{\theta_i})_{i\le n}$.
- **Output:** for $i = 1$ to n, the sender outputs $a_i \leftarrow H_i^\rho$, and the receiver outputs $\alpha_i \leftarrow C_1^{-z_i} \cdot C_0^{\theta_i}$.

Observe that

$$\alpha_i = C_1^{-z_i} \cdot C_0^{\theta_i} = G^{\rho\theta_i} \cdot H^{-\rho z_i} r^{-z_i} = a_i \cdot r^{-z_i}.$$

Furthermore, r is computationally indistinguishable from a random element of $\mathbb{G}'$ under the DDH assumption over $\mathbb{G}'$, and the protocol statistically hides z_i.

A first downside of this protocol is that we cannot set $\mathbb{Z}_p^* = \mathbb{G}'$, since DDH is easy over $\mathbb{Z}_p^*$ (it can be broken by computing the Legendre symbol). However, assuming that $p = 2q + 1$ is a safe prime (q is prime), we can set $\mathbb{G}'$ to be the subgroup QR_p of quadratic residues modulo p, where DDH is widely conjectured to hold (for a sufficiently large p). This implies that the protocol generates a (pseudo)random *square* r, instead of a random element of $\mathbb{Z}_p^*$. This does not harm the security of the CPRF but changes slightly the underlying sparse power-DDH variant: using r of the form w^2 for a (pseudo)random element w when computing $g_t \leftarrow g^{r^t} = g^{w^{2t}}$ for $t \in S'$ amounts exactly to relying on the sparse power-DDH assumption with respect to the set $2 \cdot S' = \{2 \cdot t \ : \ t \in S'\}$.

A second, more concerning, downside is the size of p: due to subexponential-time algorithms for discrete logarithm over finite fields, p should be taken much larger than 256 bits, at the very least 1024 bits. But in turn, this implies that the group $\mathbb{G}$ over which we instantiate our PCF should have order $p \ge 2^{1024}$, which considerably harms efficiency (both for key size and computation), and prevents us in particular to rely on efficient 256-bit elliptic curves. We circumvent this issue by setting p to a smaller value (e.g. a 256-bit prime), and relying on Paillier encryption.

A more efficient variant. Assume for simplicity that $p = 2q+1$ for a prime q (the construction also works fine with any large prime factor of $p - 1$). At a high level, we perform the Bellare-Micali-style non-interactive protocol over a Paillier group (similarly as in [OSY21]) followed by a post-processing operation which:

- converts the multiplicative shares over the Paillier group to subtractive shares modulo N (where N is an RSA modulus) using a distributed discrete log algorithm,
- converts the shares modulo N to shares modulo q using the fact that subtractive shares modulo N are with very high probability shares over $\mathbb{Z}$ when the shared value is sufficiently smaller than the modulus,

– converts the additive shares modulo q into multiplicative shares over $\mathbb{Z}_p^*$ via exponentiation.

Let QR_p denote the set of quadratic residues modulo p, which has order q. Let G be a basis of QR_p. Instead of sampling $r \xleftarrow{\$} \mathsf{QR}_p$ directly, Alice samples $\Delta \xleftarrow{\$} \mathbb{Z}_q$ and sets $r \leftarrow G^\Delta \bmod p$ (this yields the same distribution). Let N be a public RSA modulus, whose factorization is unknown to both parties. The protocol proceeds almost as the previous protocol, except that Alice sends a Paillier-ElGamal encryption of Δ (viewed as an integer in $\{0, \cdots, q-1\}$) instead of an ElGamal encryption of r. Let $(\mathbf{G}, \mathbf{H})$ be two random elements of $\mathbb{Z}_{N^2}$. Our protocol borrows ideas from [OSY21]. It builds upon a *distributed discrete logarithm* algorithm DDLOG over $\mathbb{Z}_{N^2}$, which has the following features: given respective multiplicative shares $(S^{\mathsf{send}}, S^{\mathsf{rec}})$ of a value $(1+N)^m$ modulo N^2, the sender and the receiver can locally compute $v^{\mathsf{send}} \leftarrow \mathsf{DDLOG}(\mathsf{send}, S^{\mathsf{send}})$ and $v^{\mathsf{rec}} \leftarrow \mathsf{DDLOG}(\mathsf{rec}, S^{\mathsf{rec}})$ which form subtractive shares of m over $\mathbb{Z}_N$ (*i.e.* $v^{\mathsf{send}} - v^{\mathsf{rec}} = m \bmod N$). Furthermore, if $m < N/2^\lambda$ (when viewed as an integer in $\{0, \cdots, N-1\}$), it holds with probability at least $1 - 2^{-\lambda}$ that $v^{\mathsf{send}} - v^{\mathsf{rec}} = m$ *over the integers*. The work of [OSY21] described an efficient implementation of DDLOG, whose cost boils down to one inversion and one multiplication over $\mathbb{Z}_N$. Given this procedure, our protocol proceeds as follows:

– **Sender to receiver:** pick a random coin ρ, and sends the Paillier-ElGamal ciphertext $(C_0, C_1) \leftarrow (\mathbf{G}^\rho \bmod N, \mathbf{H}^\rho \cdot (1+N)^\Delta \bmod N^2)$.
– **Receiver to sender:** pick n random coins $(\theta_1, \cdots, \theta_n)$ and send the Pedersen commitments $(H_1, \cdots, H_n) \leftarrow (\mathbf{H}^{z_i} \cdot \mathbf{G}^{\theta_i} \bmod N^2)_{i \le n}$.
– **Output:** for $i = 1$ to n, the sender computes $\mathbf{G}_i^{\mathsf{send}} \leftarrow H_i^\rho$, and the receiver computes $\mathbf{G}_i^{\mathsf{rec}} \leftarrow C_1^{z_i} \cdot C_0^{\theta_i}$. Observe that

$$\mathbf{G}_i^{\mathsf{rec}} = C_1^{z_i} \cdot C_0^{\theta_i} = G^{\rho\theta_i} \cdot H^{\rho z_i} (1+N)^{\Delta \cdot z_i} = \mathbf{G}_i^{\mathsf{send}} \cdot (1+N)^{\Delta \cdot z_i} \bmod N^2.$$

Using DDLOG, both parties locally compute values $(v_i^{\mathsf{send}}, v_i^{\mathsf{rec}})$ such that $v_i^{\mathsf{send}} - v_i^{\mathsf{rec}} = b \cdot \Delta \bmod N$. Furthermore, assuming that $q < N/2^\lambda$,[9] it holds that $v_i^{\mathsf{send}} - v_i^{\mathsf{rec}} = b \cdot \Delta$ over $\mathbb{Z}$ with probability at least $1 - 1/2^\lambda$. Eventually, the sender outputs $a_i \leftarrow G^{v_i^{\mathsf{send}}}$ and the receiver outputs $\alpha_i \leftarrow G^{v_i^{\mathsf{rec}}}$. Observe that

$$a_i = G^{v_i^{\mathsf{send}}} = G^{v_i^{\mathsf{rec}} + b \cdot \Delta} = \alpha_i \cdot r^b \bmod p.$$

A balancing optimization. We note that in the above protocol, the size of the public keys is quite unbalanced: the sender public key contains a single Paillier-ElGamal ciphertext (in addition to $(g_t)_{t \in S'}$), while the receiver public key contains n Pedersen commitments over $\mathbb{Z}_{N^2}$ (where n is the wPRF key length, e.g. $n = 256$ for our XOR-MAJ candidate, or $n = 770$ for our BIPSW candidate). We now describe an optimization which reduces the receiver key size by a factor k, at the cost of increasing the Paillier-ElGamal ciphertext by a factor k^2. Taking $k = O(n^{1/3})$, this yields a variant in which both public keys contain $O(n^{2/3})$ elements of $\mathbb{Z}_{N^2}$. We note that our balancing optimization also applies to the public key PCF of [OSY21], thus enables reducing their public key size to $O(n^{2/3})$.

The main idea of the optimization is to compress the receiver public key by replacing the Pedersen commitments with a multi-Pedersen commitment. Fix a compression parameter k (which we assume to divide n for simplicity) and public random elements $(\mathbf{G}, \mathbf{H}_1, \cdots, \mathbf{H}_k) \in \mathbb{Z}_{N^2}^{k+1}$. We let the sender commits to z by batches of k values z_i at once, as follows:

– **Receiver to sender:** pick n/k random coins $(\theta_1, \cdots, \theta_{n/k})$ and send the Pedersen commitments $(H_1, \cdots, H_{n/k}) \leftarrow (\mathbf{G}^{\theta_{i+1}} \cdot \prod_{j=1}^k \mathbf{H}_j^{z_{j+k \cdot i}} \bmod N^2)_{0 \le i < n/k}$.

Suppose the parties want to retrieve multiplicative shares of $(1+N)^{\Delta \cdot z_i} \bmod N^2$. The main observation is that this can be done using the randomness-reuse variant of Paillier-ElGamal, putting $(1+N)^\Delta$ in the first "slot": the sender picks a random coin ρ_1 and computes

$$(C_0, (C_1^j)_{j \le k}) \leftarrow (\mathbf{G}^{\rho_1} \bmod N, \mathbf{H}_1^{\rho_1} \cdot (1+N)^\Delta, \mathbf{H}_2^{\rho_1}, \cdots, \mathbf{H}_k^{\rho_1} \bmod N^2).$$

Then, given this extended ciphertext and $H_1 = \mathbf{G}^{\theta_1} \cdot \prod_{j=1}^k \mathbf{H}_j^{z_j} \bmod N^2$, the parties retrieve multiplicative shares of $(1+N)^{\Delta \cdot z_1}$ by computing

$$\mathbf{G}_1^{\mathsf{send}} \leftarrow H_1^{\rho_1}, \qquad \mathbf{G}_1^{\mathsf{rec}} \leftarrow C_0^{\theta_1} \cdot \prod_{j \le k} (C_1^j)^{z_j}.$$

[9] In practice, we take $\log q = 256$ and $\log N = 3072$.

The above only yields shares of $(1+N)^{\Delta \cdot z_1}$. To extract shares of $(1+N)^{\Delta \cdot z_j}$ for $j = 2, \cdots, k$, the sender must proceed similarly as above, using extended Paillier-ElGamal ciphertexts, but this time placing $(1+N)^{\Delta}$ in the j-th slot. In total, the sender computes k length-$(k+1)$ extended ciphertexts, for a total of k elements of $\mathbb{Z}_N$ and k^2 elements of $\mathbb{Z}_{N^2}$ (these $k + k^2$ elements can be reused across all n/k batches). The full sender public key is given below:

– **Sender to receiver:** pick random coins ρ_j for $j = 1$ to k, and constructs the extended Paillier-ElGamal ciphertexts as follows for $j = 1$ to k:

$$C_0^j \leftarrow \mathbf{G}^{\rho_j} \bmod N$$
$$(C_1^{j,1} \cdots, C_1^{j,k}) \leftarrow (\mathbf{H}_1^{\rho_j}, \cdots, \mathbf{H}_j^{\rho_j} \cdot (1+N)^{\Delta}, \cdots, \mathbf{H}_k^{\rho_j}) \bmod N^2$$

Efficiency. Without the balancing optimization, the public key of the sender consists in $|S'|$ elements of $\mathbb{G}$, one element of $\mathbb{Z}_N$, and one element of $\mathbb{Z}_{N^2}$, and the public key of the receiver consists in n elements of $\mathbb{Z}_{N^2}$. To provide concrete estimates, we use our XOR-MAJ parameter set with $n = 256$ and $|S'| = 390$. We set $\lambda = 128$, $\log|\mathbb{G}| = 2\lambda$, and $\log N = 3072$. With these parameters, the sender public key size is 13.3kB, and the receiver public key size is 192kB. Using the balancing optimization, the public key of the sender consists in $|S'|$ elements of $\mathbb{G}$, k element of $\mathbb{Z}_N$, and k^2 element of $\mathbb{Z}_{N^2}$, and the public key of the receiver consists in n/k elements of $\mathbb{Z}_{N^2}$. With the XOR-MAJ parameter set and using $k = 5$, the sender key increases to 32.8kB while the receiver key is reduced to 38.4kB.

Regarding computation, the cost of deriving the PCF keys from the public and secret keys is dominated by $n+1$ exponentiations modulo N^2 and n exponentiations modulo p for the sender, and $2n$ exponentiations modulo N^2 and n exponentiations modulo p for the receiver. Using the balancing optimization, the number of exponentiations modulo N^2 increases to $n + k^2$ for the sender, and decreases to $n \cdot (1 + 1/k)$ for the receiver. Using $n = 256$ and $k = 5$, this translates to respectively 281 and 307 exponentiations over $\mathbb{Z}_{N^2}$.

Using $\log N = 3072$, an exponentiation modulo N^2 takes of the order of 5ms on one core a standard laptop, which translates to $1 \sim 2$ seconds of computation (note that this is a rough back-of-the-envelope estimation, true estimates may vary). Observe that this can be easily sped up using multiple cores, and that this is a one-time preprocessing phase to generate the shared PCF keys. After generating the PCF keys once for all, the parties can directly start generating OT correlations. Furthermore, the computational efficiency can be significantly improved by sampling ρ and the θ_i's as 256-bit integers. This improves computation by one to two orders of magnitude, at the (reasonable) cost of having to assume the security of the small-exponent indistinguishability assumption (see e.g. [CC18, CKLR21] for discussions on this assumption and relations to other assumptions).

2.8 Application: A simple reusable DV-NIZK reusable

As a final contribution, we provide a way to use our PK-PCF in order to construct *reusable* DV-NIZKs from three ingredients: (1) A Σ-protocol [CDS94] with 1-bit challenges for an NP-complete language $\mathcal{L}$, for example Blum's protocol for graph Hamiltonicity [Blu86], (2) a public key PCF for OT correlation where the key evaluation of each party can be *silently* obtained from their own secret key and public key of the other, and (3) a *non-reusable* DV-NIZK with computational adaptive soundness knowledge and adaptive zero-knowledge properties. Note that this last ingredient can be constructed from public-key encryption and λ invocations of Σ-protocol [PsV06].

The main idea behind our construction is the following. The designated verifier samples a PCF key pair $(\mathsf{sk}_V, \mathsf{pk}_V)$ and outputs a CRS containing their public key. A prover with statement x and witness w can then sample their own PCF key pair $(\mathsf{sk}_P, \mathsf{pk}_P)$ to produce a shared evaluation key with the designated prover. It then runs the Σ-protocol by computing a first message a. The challenge being binary, there are 2 possible third message for a transcript starting with a. We let z_b the third message for challenge $b \in \{0,1\}$. Doing this λ-times lead to 2λ triplets $(a_i, b, z_{i,b})_{i\in[\lambda], b\in\{0,1\}}$. The prover then uses their PCF evaluation key to compute 2λ pseudorandom masks $r_{i,b}$ by evaluating the PCF on input $x|i|b$ (or $H(x|i|b)$ if the PCF is only weakly-secure). The prover finally outputs $\mathsf{pk}_P, (a_i, z_{i,0} \oplus r_{i,0}, z_{i,1} \oplus r_{i,1})$ as their proof.

The correctness of the PCF and Σ-protocol guarantee that the designated verifier can recover 1 mask out of each pair $(r_{i,0}, r_{i,1})$ and then can verify λ-transcripts, while security guarantees that the prover cannot predict which of the two is recovered by the verifier and that the non-revealed

$r_{i,b}$ is pseudorandom, therefore providing soundness and zero-knowledge. A minor issue remains: one needs to prevent the prover to sample maliciously their PCF key pair such that it can predict the challenge bit. We show that it is sufficient to require the prover to additionally provide a proof (using a non-reusable DV-NIZK) that their PCF public key was generated from a honest execution of the PCF key generation algorithm (with possibly bad randomness).

3 Preliminaries

We use λ to denote the security parameter. For a natural integer $n \in \mathbb{N}$, the set $\{1, \ldots, n\}$ is denoted by $[n]$. We mostly use bold lowercase letters (e.g., $\mathbf{r}$) to denote vectors. For a vector $\mathbf{r} = (r_1, \ldots, r_n)$, the vector $(g^{r_1} \ldots, g^{r_n})$ is sometimes denoted by $g^{\mathbf{r}}$. We write $\mathsf{poly}(\lambda)$ to denote an arbitrary polynomial function. We denote by $\mathsf{negl}(\lambda)$ a negligible function in λ, and PPT stands for probabilistic polynomial-time. For a finite set S, we write $x \xleftarrow{\$} S$ to denote that x is sampled uniformly at random from S. For an algorithm $\mathcal{A}$, we denote by $y \leftarrow \mathcal{A}(x)$ the output y after running $\mathcal{A}$ on input x. We consider GenPar as a PPT algorithm that on input 1^λ for $\lambda \in \mathbb{N}$, outputs $(\mathbb{G}, g, p)$, where $\mathbb{G}$ is a cyclic group of prime order p generated by g.

3.1 Pseudorandom Functions

Definition 1 ((Weak) Pseudorandom Function (wPRF, PRF), [GGM84a, NR95]). Let $\lambda \in \mathbb{N}$ be a security parameter. A (weak) pseudorandom function with domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$, key space $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda\in\mathbb{N}}$, and range $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$, consists of the following two polynomial-time algorithms:

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{msk})$: A probabilistic algorithm that on input the security parameter λ, outputs a master secret key $\mathsf{msk} \in \mathcal{K}$.
- $\mathsf{Eval}(\mathsf{msk}, x) \to y$: A deterministic algorithm that on input the master secret key msk, and an input value $x \in \mathcal{X}$, outputs a value $y \in \mathcal{Y}$.

We say that the pair $(\mathsf{KeyGen}, \mathsf{Eval})$ is a

- **pseudorandom function (PRF)** if for any PPT adversary $\mathcal{A}$, it holds that

$$\left|\Pr\left[\mathcal{A}^{\mathsf{Eval}(\mathsf{msk},\cdot)}(1^\lambda) = 1 \middle| \mathsf{msk} \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)\right] - \Pr\left[\mathcal{A}^{RF(\cdot)}(1^\lambda) = 1 \middle| RF \xleftarrow{\$} \mathcal{F}\right]\right| = \mathsf{negl}(\lambda),$$

where $\mathcal{F}$ is the set of all functions with domain $\mathcal{X}$ and range $\mathcal{Y}$.

- **weak pseudorandom function (wPRF)** if for any PPT adversary $\mathcal{A}$ and any polynomially bounded number $Q \in \mathbb{N}$, it holds that

$$\left\{\left((x_i, \mathsf{Eval}(\mathsf{msk}, x_i))_{i\in[Q]}\right) \middle| \begin{array}{c}\mathsf{msk} \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda) \\ \forall i \in [Q] : x_i \xleftarrow{\$} \mathcal{X}\end{array}\right\} \approx_c \left\{\left((x_i, y_i)_{i\in[Q]}\right) \middle| \begin{array}{c}\forall i \in [Q] : \\ x_i \xleftarrow{\$} \mathcal{X},\ y_i \xleftarrow{\$} \mathcal{Y}\end{array}\right\}.$$

3.2 Constrained Pseudorandom Functions

Definition 2 (Constrained Pseudorandom Functions). Let λ be a security parameter. A Constrained Pseudorandom Function (CPRF) with domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$, key space $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda\in\mathbb{N}}$, and range $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$, that supports a class of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$, where each $C_\lambda \in \mathcal{C}_\lambda$ has domain $\mathcal{X}_\lambda$ and range $\{0, 1\}$, consists of the following four polynomial-time algorithms:[10]

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pp}, \mathsf{msk})$: The master key generation algorithm is a probabilistic algorithm that on input the security parameter λ, outputs a public parameter pp and a master secret key $\mathsf{msk} \in \mathcal{K}$.
- $\mathsf{Eval}(\mathsf{pp}, \mathsf{msk}, x) \to y$: The evaluation algorithm is a deterministic algorithm that on input the public parameter pp, the master secret key msk, and an input $x \in \mathcal{X}$, outputs a value $y \in \mathcal{Y}$.

[10] In the rest of the paper, we drop the subscript λ when it is clear from context.

- $\mathsf{Constrain}(\mathsf{msk}, C) \to \mathsf{ck}_C$: The constrained key generation algorithm is a probabilistic algorithm that on input the master secret key msk, and a circuit $C \in \mathcal{C}$, outputs a constrained key ck_C.
- $\mathsf{CEval}(\mathsf{pp}, \mathsf{ck}_C, x) \to y$: The constrained evaluation algorithm is a deterministic algorithm that on input the public parameter pp, a constrained key ck_C, and an input $x \in \mathcal{X}$, outputs a value $y \in \mathcal{Y}$.

Correctness. For any security parameter λ, any constrain $C \in \mathcal{C}$, and any input $x \in \mathcal{X}$ such that $C(x) = 0$, we have:

$$\Pr\left[\mathsf{Eval}(\mathsf{pp}, \mathsf{msk}, x) \neq \mathsf{CEval}(\mathsf{pp}, \mathsf{ck}_C, x) : \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{msk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathsf{ck}_C \leftarrow \mathsf{Constrain}(\mathsf{msk}, C) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

1-Key Selective Security. We say that a CPRF is 1-key selectively secure if the advantage of any PPT adversary $\mathcal{A}$ in the following game is negligible:

- **Setup:** The challenger runs $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, initializes a set $S_{\mathsf{eval}} = \varnothing$, and chooses a random bit $b \xleftarrow{\$} \{0,1\}$. It then sends pp to $\mathcal{A}$.
- **Selective Choice of Constraint:** The adversary chooses a (single) circuit $C \in \mathcal{C}$ and sends it to the challenger.
- **Constrained Key Generation:** The challenger computes $\mathsf{ck}_C \leftarrow \mathsf{Constrain}(\mathsf{msk}, C)$ and returns the constrained key ck_C to $\mathcal{A}$.
- **Pre-Challenge Evaluation Queries:** $\mathcal{A}$ can adaptively send arbitrary input values $x \in \mathcal{X}$ to the challenger. The challenger computes $y \leftarrow \mathsf{Eval}(\mathsf{pp}, \mathsf{msk}, x)$ and returns y to $\mathcal{A}$. It also updates $S_{\mathsf{eval}} \leftarrow S_{\mathsf{eval}} \cup \{x\}$.
- **Challenge Phase:** $\mathcal{A}$ sends an input $x^* \in \mathcal{X}$ as its challenge query to the challenger with the restriction that $x^* \notin S_{\mathsf{eval}}$ and $C(x^*) \neq 0$. If it holds that $b = 0$, then the challenger computes $y^* \leftarrow \mathsf{Eval}(\mathsf{pp}, \mathsf{msk}, x^*)$. Otherwise, if $b = 1$, the challenger samples a random value $y^* \xleftarrow{\$} \mathcal{Y}$. Finally, the challenger returns y^* to $\mathcal{A}$.
- **Post-Challenge Evaluation Queries:** $\mathcal{A}$ continues the queries as before, with the restriction that it cannot query x^* as an evaluation query.
- **Guess:** $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.

1-Key Selective Constraint-Hiding. We say that a CPRF is selectively 1-key constraint-hiding if the advantage of any PPT adversary $\mathcal{A}$ in the following game is negligible:

- **Setup:** The challenger runs $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and chooses a random bit $b \xleftarrow{\$} \{0,1\}$. It then sends pp to $\mathcal{A}$.
- **Selective Choice of Constraint:** The adversary chooses a (single) pair of circuits $(C_0, C_1) \in \mathcal{C}$ and sends the pair to the challenger.
- **Constrained Key Generation:** The challenger computes $\mathsf{ck}_b \leftarrow \mathsf{Constrain}(\mathsf{msk}, C_b)$, and returns ck_b to $\mathcal{A}$.
- **Evaluation Queries:** $\mathcal{A}$ can query the output of the evaluation algorithm on arbitrary inputs $x \in \mathcal{X}$, with the restriction that $C_0(x) = C_1(x)$. On such inputs, the challenger computes and returns $y \leftarrow \mathsf{Eval}(\mathsf{pp}, \mathsf{msk}, x)$ to $\mathcal{A}$.
- **Guess:** $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.

In both of the games described above, $\mathcal{A}$ wins if $b' = b$. We also define the advantage of $\mathcal{A}$ in winning a game as $|2 \cdot \Pr[\mathcal{A} \text{ wins}] - 1|$, where the probability is over the internal coins of $\mathcal{A}$ and the challenger.

No-Evaluation Security. 1-key selective no-evaluation security (resp. 1-key selective no-evaluation constraint-hiding) is defined similarly with the extra restriction that the adversary cannot issue any pre-challenge or post-challenge query (resp. any evaluation query).

3.3 Reverse-Sampleable Correlations

Definition 3 (Reverse-Sampleable Correlation). *Let $1 \leq \ell_0(\lambda), \ell_1(\lambda) \leq \mathsf{poly}(\lambda)$ be output-length functions. Let $\mathcal{Y}$ be a probabilistic algorithm that, 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 probabilistic polynomial time 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 $\sigma \in \{0,1\}$ the following distributions are statistically close:*

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

Definition 4 (OT Correlation). *A* (1-out-of-2, bit) OT correlation *can be defined as being sampled as a pair $((r_0, r_1), (b, r_b))$, where $r_0, r_1, b \xleftarrow{\$} \{0,1\}$.*

Remark 1 (An OT Correlation is Reverse-Sampleable). *A (1-out-of-2, bit) OT correlation is reverse-sampleable.* Indeed, observe that the reverse-sampling can be performed as follows. $\mathsf{RSample}(1^\sigma, \sigma, y_\sigma)$: If $\sigma = 0$, parse y_σ as $y_\sigma = (r_0, r_1)$, sample $b \xleftarrow{\$} \{0,1\}$, and output (b, r_b); otherwise (*i.e.* if $\sigma = 1$) parse y_σ as $y_\sigma = (b, r)$, sample $r' \xleftarrow{\$} \{0,1\}$, and output $((1-b)\cdot r + b \cdot r', b \cdot r + (1-b)\cdot r')$.

3.4 Pseudorandom Correlation Functions

At a high level, a *pseudorandom correlation function* (PCF) compresses, in short correlated keys, (superpolynomially large) correlated pseudorandom strings for some ideal correlation, *e.g.* strings of Beaver triples [Bea92][11]. For instance, a key owner can evaluate the PCF at position i to recover its share of the i^{th} Beaver triple.

We consider two different flavours of PCFs: *weak* PCFs (wPCF) and *strong* PCFs (sPCF). Analogously to PRFs, wPCFs guarantee security given access only to evaluations on uniformly random and independent inputs, while sPCFs guarantee security even for adaptively chosen inputs. Note that contrary to PRFs, the PCF literature treats *weak* PCFs as the default notion.

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 3). We refer to [BCG+20, Section 4] for more details.

Weak Pseudorandom Correlation Functions (wPCF). We start by defining the notion of a *weak* pseudorandom correlation function.

Definition 5 ((Weak) Pseudorandom Correlation Function (wPCF), [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. Let* $(\mathsf{wPCF.Gen}, \mathsf{wPCF.Eval})$ *be a pair of algorithms 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 $y_\sigma \in \{0,1\}^{\ell_\sigma(\lambda)}$.*

We say that $(\mathsf{wPCF.Gen}, \mathsf{wPCF.Eval})$ *is a pseudorandom correlation function (PCF) for $\mathcal{Y}$, if the following conditions hold:*

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

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

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

[11] Recall that a Beaver triple is a triplet additive shares $([a], [b], [c])$ where $a, b \xleftarrow{\$} \mathcal{R}$ for some ring $\mathcal{R}$, and $c = ab$.

Experiment (Weakly) Pseudorandom Correlated Outputs

$\underline{\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},N,0}(\lambda)}$:

$(k_0, k_1) \leftarrow \mathsf{PCF.Gen}(1^\lambda)$
For $i = 1, \ldots, N(\lambda)$:
$x^{(i)} \xleftarrow{\$} \{0,1\}^{n(\lambda)}$
$(y_0^{(i)}, y_1^{(i)}) \hookleftarrow \mathcal{Y}(1^\lambda)$
$b \leftarrow \mathcal{A}(1^\lambda, (x^{(i)}, y_0^{(i)}, y_1^{(i)})_{i\in[N(\lambda)]})$
Output b

$\underline{\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},N,1}(\lambda)}$:

$(k_0, k_1) \leftarrow \mathsf{PCF.Gen}(1^\lambda)$
For $i = 1, \ldots, N(\lambda)$:
$x^{(i)} \xleftarrow{\$} \{0,1\}^{n(\lambda)}$
For $\sigma \in \{0,1\}$**:**
$y_\sigma^{(i)} \xleftarrow{\$} \mathsf{wPCF.Eval}(\sigma, k_\sigma, x^{(i)})$
$b \leftarrow \mathcal{A}(1^\lambda, (x^{(i)}, y_0^{(i)}, y_1^{(i)})_{i\in[N(\lambda)]})$
Output b

Fig. 1. *(Weakly) Pseudorandom $\mathcal{Y}$-correlated outputs of a (w)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 2. In particular, the adversary is given access to $N(\lambda)$ samples (or simply N if there is no ambiguity).

Experiment (Weak) PCF Security

$\underline{\mathsf{Exp}^{\mathsf{w\text{-}sec}}_{\mathcal{A},N,\sigma,0}(\lambda)}$:

$(k_0, k_1) \leftarrow \mathsf{PCF.Gen}(1^\lambda)$
For $i = 1, \ldots, N(\lambda)$:
$x^{(i)} \xleftarrow{\$} \{0,1\}^{n(\lambda)}$
$y_\sigma^{(i)} \xleftarrow{\$} \mathsf{wPCF.Eval}(\sigma, k_\sigma, x^{(i)})$
$y_{1-\sigma}^{(i)} \xleftarrow{\$} \mathsf{RSample}(1^\lambda, \sigma, y_\sigma^{(i)})$
*
$b \leftarrow \mathcal{A}(1^\lambda, \sigma, k_\sigma, (x^{(i)}, y_{1-\sigma}^{(i)})_{i\in[N(\lambda)]})$
Output b

$\underline{\mathsf{Exp}^{\mathsf{w\text{-}sec}}_{\mathcal{A},N,\sigma,1}(\lambda)}$:

$(k_0, k_1) \leftarrow \mathsf{PCF.Gen}(1^\lambda)$
For $i = 1, \ldots, N(\lambda)$:
$x^{(i)} \xleftarrow{\$} \{0,1\}^{n(\lambda)}$
$y_{1-\sigma}^{(i)} \xleftarrow{\$} \mathsf{wPCF.Eval}(1-\sigma, k_{1-\sigma}, x^{(i)})$
$b \leftarrow \mathcal{A}(1^\lambda, \sigma, k_\sigma, (x^{(i)}, y_{1-\sigma}^{(i)})_{i\in[N(\lambda)]})$
Output b

Fig. 2. *Security of a wPCF.* $\mathsf{RSample}$ *is the algorithm for reverse sampling $\mathcal{Y}$ as in Definition 3.*

Strong Pseudorandom Correlation Functions. A strong PCF is syntactically defined in the same way as a weak PCF, but it instead satisfies stronger notions of *pseudorandom $\mathcal{Y}$-correlated outputs* and *PCF security*. For simplicity, we only provide these modified properties.

We say that $(\mathsf{sPCF.Gen}, \mathsf{sPCF.Eval})$ is an (N, B, ϵ)-secure strong pseudorandom correlation function (sPCF) for $\mathcal{Y}$, if the following conditions hold:

– **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 3), it holds that for all sufficiently large λ,

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

where $\mathsf{Exp}^{\mathsf{s\text{-}pr}}_{\mathcal{A},b}$ ($b \in \{0,1\}$) is defined as in Figure 3.

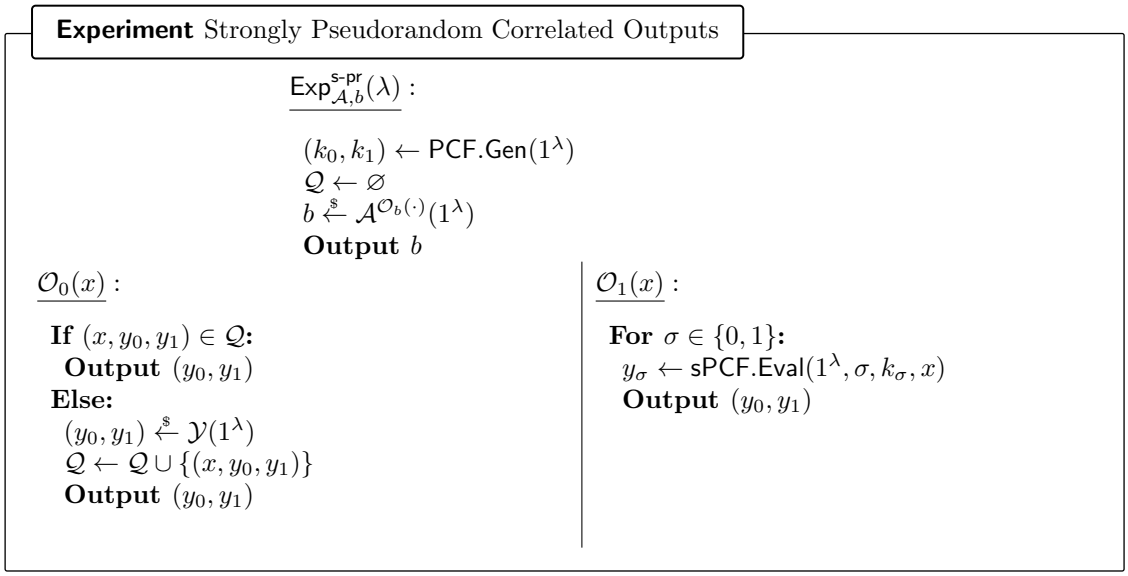

Fig. 3. Strongly Pseudorandom $\mathcal{Y}$-correlated outputs of a sPCF.

– **Strong 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 4), 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 4.

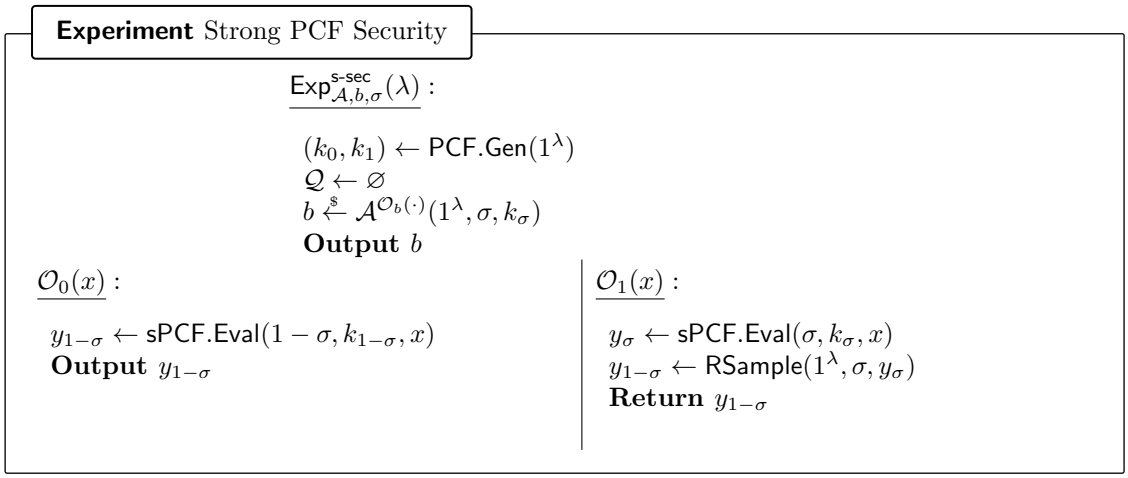

Fig. 4. Security of a strong PCF. Here, $\mathsf{RSample}$ is the algorithm for reverse sampling $\mathcal{Y}$ as in Definition 3.

At a high level, a PCF is *precomputable* if the first party's key can be generated first, and the second key can be derived from the first.

Definition 6 (Precomputable Pseudorandom Correlation Function, [CMPR23]). Let $\mathcal{Y}$ be a reverse-sampleable correlation with output lengths $\ell_0(\lambda), \ell_1(\lambda)$ and let $\lambda \leq n(\lambda) \leq \mathsf{poly}(\lambda)$ be its input length. We say that a pseudorandom correlation function $(\mathsf{PCF.Gen}, \mathsf{PCF.Eval})$ is *precomputable* if the description of $\mathsf{PCF.Gen}$ contains the descriptions of two algorithms $(\mathsf{PCF.Gen}_0, \mathsf{PCF.Gen}_1)$ such that

- $\mathsf{PCF.Gen}_0(1^\lambda)$: On input the security parameter λ, returns a key k_0 and auxiliary output aux.
- $\mathsf{PCF.Gen}_1(1^\lambda, \mathsf{aux})$: On input the security parameter λ and an auxiliary input aux, outputs a key k_1.

We also require the following property to hold:
Precomputability. For any security parameter $\lambda \in \mathbb{N}$, the two following distributions are computationally indistinguishable:

$$\left\{(k_0,k_1)\colon (k_0,k_1) \leftarrow \mathsf{PCF.Gen}(1^\lambda)\right\} \overset{c}{\approx} \left\{(k_0,k_1)\colon \begin{array}{c}(k_0,\mathsf{aux}) \leftarrow \mathsf{PCF.Gen}_0(1^\lambda)\\ k_1 \leftarrow \mathsf{PCF.Gen}_1(1^\lambda,\mathsf{aux})\end{array}\right\}.$$

3.5 NIZKs

A non-interactive argument for $\mathcal{R}$ is a tuple of three probabilistic polynomial time interactive algorithms $\Pi = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ called the common reference string generator, the prover, and the verifier with the following properties:

- $\mathsf{Setup}(1^\lambda)$. On input 1^λ generates public parameters *par* (such as group parameters), a crs, and a trapdoor $\mathcal{T}$. For simplicity of notation, we assume that any group parameters are implicitly included in the crs.
- $\mathsf{P}(\mathsf{crs}, x, w)$. On input of a crs, a statement x with witness w, outputs a proof π for $x \in \mathscr{L}$.
- $\mathsf{V}(\mathsf{crs}, x, \pi, \mathcal{T})$. On input of a crs, a statement, a proof, and a trapdoor, accepts or rejects the proof.

which satisfies the completeness, soundness, and zero-knowledge properties defined below.

If the trapdoor $\mathcal{T}$ of the non-interactive proof system is set to $\perp$ (or, alternatively, if it is included in the crs), we call the argument system *publicly verifiable*. Otherwise, we call it a *designated-verifier* non-interactive argument system. If the soundness guarantee holds with respect to a computationally unbounded adversary, we have a NIZK-proof system.

Definition 7 (Perfect completeness). *A proof system $\Pi = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ for $\mathcal{R}$ is perfectly complete, if*

$$\Pr\left[\mathsf{V}(\mathsf{crs}, x, \pi, \mathcal{T}) = 1 \;\middle|\; \begin{array}{c}(\mathsf{crs}, \mathcal{T}) \leftarrow \mathsf{Setup}(1^\lambda)\\ (x,w) \in \mathcal{R}\ ,\ \pi \xleftarrow{\$} \mathsf{P}(\mathsf{crs}, x, w)\end{array}\right] = 1$$

The soundness notion can be divided into *non-adaptive* and *adaptive*; it is non-adaptive if the malicious prover needs to choose the statement x before generating the crs while it is adaptive if the adversary can dynamically choose the statement after generating crs. We consider a strong variant of adaptive soundness, denoted *unbounded adaptive* soundness, where the adversary is given oracle access to a verification oracle. Note that in the publicly-verifiable setting, this is equivalent to the standard soundness notion (computational soundness), where the adversary must forge valid proof on an incorrect statement without the help of any oracle. However, in the designated-verifier setting, the standard soundness notion only guarantees that the argument system remains sound as long as the prover receives at most logarithmically responses on previous proofs. On the other hand, if the argument system satisfies unbounded soundness (*reusable soundness*), its soundness is maintained even if the adversary receives an arbitrary (polynomial) number of responses on previous proofs.

Definition 8 (Unbounded adaptive soundness). *A proof system Π is unbounded adaptive soundness if for every PPT adversary $\mathcal{A}$*

$$\Pr\left[\mathsf{V}(\mathsf{crs}, x, \pi, \mathcal{T}) = 1 \;\middle|\; \begin{array}{c}(\mathsf{crs}, \mathcal{T}) \leftarrow \mathsf{Setup}(1^\lambda)\\ x \notin \mathscr{L}\\ (x,\pi) \leftarrow \mathcal{A}^{\mathcal{O}(\mathsf{crs},.,.,\mathcal{T})}(\mathsf{crs})\end{array}\right] = \mathsf{negl}(\lambda)$$

where $\mathcal{A}$ can make polynomially many queries to an oracle $\mathcal{O}(\mathsf{crs}, ., ., \mathcal{T})$ which, on input (x, π), outputs $\mathsf{V}(\mathsf{crs}, x, \pi, \mathcal{T})$.

Knowledge extractability (soundness) is a strengthening of the soundness property which guarantees that if the prover produces an accepting proof then there exists an efficient simulator can actually *extract* a witness for the statement. So the extractor is defined by $\mathsf{Ext}(\pi, x, \mathcal{T}) \to w$ where $(x, w) \in \mathcal{R}$.

Definition 9 (Unbounded adaptive knowledge soundness). *A proof system Π is unbounded adaptive knowledge extractability if for every PPT adversary $\mathcal{A}$, there exists an efficient extractor* Ext *such that*

$$\Pr\left[\begin{array}{c} (\mathsf{crs},\mathcal{T}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (x,\pi) \leftarrow \mathcal{A}^{\mathcal{O}(\mathsf{crs},.,.,\mathcal{T})}(\mathsf{crs}) \;:\; (x,w) \in \mathcal{R} \text{ iff } \mathsf{V}(\mathsf{crs},x,\pi,\mathcal{T})=1 \\ w \leftarrow \mathsf{Ext}(\pi,x,\mathcal{T}) \end{array}\right] \approx 1$$

where $\mathcal{A}$ can make polynomially many queries to an oracle $\mathcal{O}(\mathsf{crs},.,.,\mathcal{T})$ which, on input (x,π), outputs $\mathsf{V}(\mathsf{crs},x,\pi,\mathcal{T})$.

We consider the notion of *adaptive* zero-knowledge where the adversary can choose the statement after seeing the crs. The definition of ZK below is often referred to as "single-theorem ZK" in which the prover generates a single proof (and the length of the common reference string can be larger than the length of the statement to prove) and multi-theorem zero-knowledge (where the adversary can adaptively ask for polynomially many proofs on arbitrary pairs (x,w) for the same common reference string). Note that, there is a generic compiler from single-theorem ZK to multi-theorem ZK where zero-knowledge holds polynomially many statements via the "OR trick". The same transformation directly applies to both the selective and adaptive ZK setting and also both the publicly verifiable and the designated verifier setting.

Definition 10 (Adaptive Zero-Knowledge). *We say a non-interactive argument Π is Adaptive Single-Theorem (Multi-Theorem) Zero-Knowledge if there exists a polynomial time simulator* $\mathsf{SimProver} = (S_1, S_2)$ *where $(\mathsf{crs},\mathcal{T}) \leftarrow S_1(1^\lambda)$ outputs a simulated common reference string and a simulation trapdoor and $\pi \leftarrow S_2(\mathsf{crs},\mathcal{T},x)$ produces a simulated argument such that*

Adaptive Single-Theorem Zero-Knowledge. For all interactive adversaries PPT $\mathcal{A}$, we require

$$\Pr\left[\begin{array}{c} \mathcal{A}(\pi)=1 \\ (x,w)\in\mathcal{R} \end{array}\middle|\begin{array}{c} (\mathsf{crs},\mathcal{T}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (x,w) \leftarrow \mathcal{A}(1^\lambda,\mathsf{crs}) \\ \pi \xleftarrow{\$} \mathsf{P}(\mathsf{crs},x,w) \end{array}\right] - \Pr\left[\begin{array}{c} \mathcal{A}(\pi)=1 \\ (x,w)\in\mathcal{R} \end{array}\middle|\begin{array}{c} (\mathsf{crs},\mathcal{T}) \leftarrow S_1(1^\lambda) \\ (x,w) \leftarrow \mathcal{A}(1^\lambda,\mathsf{crs}) \\ \pi \leftarrow S_2(\mathsf{crs},\mathcal{T},x) \end{array}\right] = \mathsf{negl}(\lambda)$$

Adaptive Multi-Theorem Zero-Knowledge. A PPT $\mathcal{A}$ has a negligible advantage in distinguishing the experiments $\mathsf{Exp}^{\mathsf{zk},0}_{\mathcal{A}}(1^\lambda)$ and $\mathsf{Exp}^{\mathsf{zk},1}_{\mathcal{A}}(1^\lambda)$ given in fig. 5.

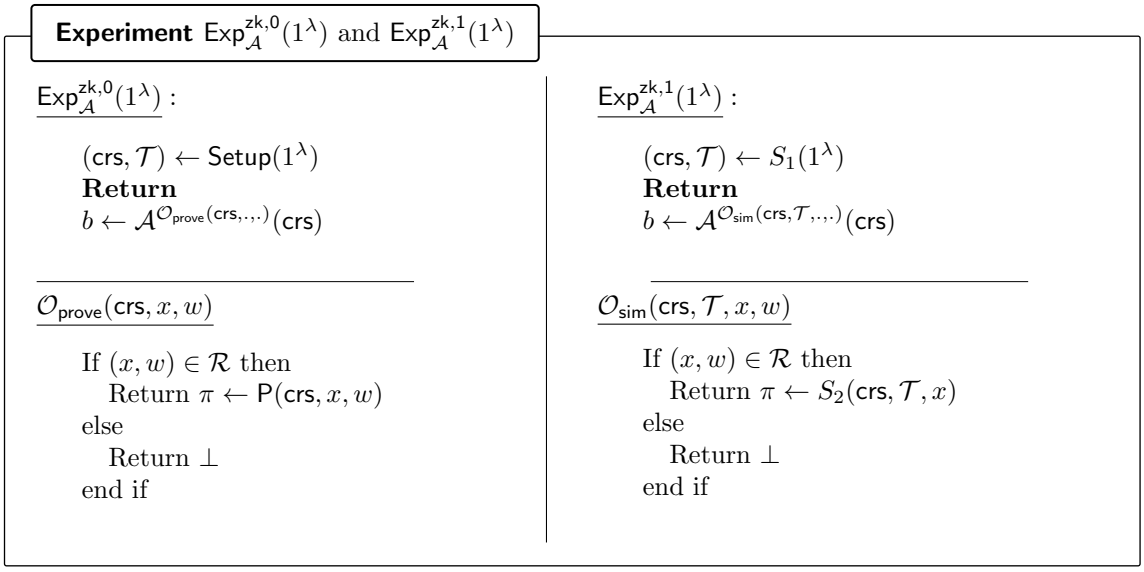

Fig. 5. $\mathsf{Exp}^{\mathsf{zk},0}_{\mathcal{A}}(1^\lambda)$ and $\mathsf{Exp}^{\mathsf{zk},1}_{\mathcal{A}}(1^\lambda)$ and oracles $\mathcal{O}_{\mathsf{prove}}(\mathsf{crs},x,w)$ and $\mathcal{O}_{\mathsf{sim}}(\mathsf{crs},\mathcal{T},x,w)$, for the (adaptive) multi-theorem zero-knowledge property of a non-interactive argument system. $\mathcal{A}$ outputs $b\in\{0,1\}$.

3.6 Variants of Power-DDH

Assumption 1 (Power-DDH, [CNs07, AHI11])
The power-DDH *assumption states that for a group $\mathbb{G}_\lambda = \langle g \rangle$ of prime order p, for any polynomially-bounded $\ell \in \mathbb{N}$, it holds that*

$$\left(g, g^r, g^{r^2}, \ldots, g^{r^{\ell-1}}, g^{r^\ell}\right) \overset{c}{\approx} \left(g, g^r, g^{r^2}, \ldots, g^{r^{\ell-1}}, g^t\right),$$

where $r, t \xleftarrow{\$} \mathbb{Z}_p^*$.

Assumption 2 (Sparse Power-DDH) *The* sparse power-DDH *assumption states that for a group $\mathbb{G}_\lambda = \langle g \rangle$ of prime order p, for any polynomially-bounded $\ell \in \mathbb{N}$ and $S \subset [\ell]$, it holds that*

$$\left(g, (g^{r^i})_{i\in S}, (g^{r^i})_{i\in[\ell]\setminus S}\right) \overset{c}{\approx} \left(g, (g^{r^i})_{i\in S}, (g^{t_i})_{i\in[\ell]\setminus S}\right),$$

where $r \xleftarrow{\$} \mathbb{Z}_p^*$, and $t_i \xleftarrow{\$} \mathbb{Z}_p^*$ for all $i \in [\ell] \setminus S$.

3.7 Decision Composite Residuosity Assumption

Let $\mathsf{SampleModulus}$ be a polynomial-time algorithm that on input the security parameter λ, outputs (N, P, Q), where $N = PQ$ for λ-bit primes P and Q.

Assumption 3 (Decision Composite Residuosity assumption, [Pai99]) Let λ be the security parameter. We say that the Decision Composite Residuosity (DCR) problem is hard relative to $\mathsf{SampleModulus}$ if $(N, x) \approx_c (N, x^N)$ where $(N, P, Q) \xleftarrow{\$} \mathsf{SampleModulus}(1^\lambda)$, $x \xleftarrow{\$} \mathbb{Z}_{N^2}^*$, and x^N is computed modulo N^2.

Note that $\mathbb{Z}_{N^2}^*$ can be written as a product of subgroups $\mathbb{H} \times \mathbb{NR}_N$, where $\mathbb{H} = \{(1+N)^i : i \in [N]\}$ is of order N, and $\mathbb{NR}_N = \{x^N : x \in \mathbb{Z}_{N^2}^*\}$ is the subgroup of N-th residues that has order $\phi(N)$.

Paillier-ElGamal Cryptosystem.
The Paillier-ElGamal cryptosystem [CS02, DGS03, BCP03] is defined by a triple $(\mathsf{PaillierEG.Gen}, \mathsf{PaillierEG.Enc}, \mathsf{PaillierEG.Dec})$, and boils down to using the ElGamal cryptosystem over the group $(\mathbb{Z}_{N^2}^\star, \times)$ where N is a Blum integer of the form $N = PQ$, where P and Q are primes:

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

1. Sample $g' \xleftarrow{\$} [N^2]$
2. Set $g \leftarrow (g')^{2N} \bmod N^2$
3. Sample $d \xleftarrow{\$} [N^2]$
4. Output $(\mathsf{pk} = g^d \bmod N^2, \mathsf{sk} = d)$

$\mathsf{PaillierEG.Enc}(\mathsf{pk}, x)$:

1. Sample $r \xleftarrow{\$} [N]$
2. Output $\mathsf{ct} = (g^r, \mathsf{pk}^r \cdot (1+N)^x)$

$\mathsf{PaillierEG.Dec}(\mathsf{sk}, \mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1))$:

1. Set $\mathsf{ct}' \leftarrow \mathsf{ct}_1 \cdot (\mathsf{ct}_0)^{-d} \bmod N^2$
2. Output $x = \frac{\mathsf{ct}'-1}{N}$

Assuming the DCR assumption (Assumption 3), the Paillier-ElGamal cryptosystem is semantically secure.

3.8 Pedersen Commitment Scheme

Let p, q be large prime numbers such that $q|p-1$. The Pedersen commitment scheme [Ped92] works over the subgroup of quadratic residues of p, denoted by QR_p as follows:

$\mathsf{Pedersen.Setup}(1^\lambda)$:
1. Sample a generator g of QR_p.
2. Sample $a \xleftarrow{\$} \mathbb{Z}_q$, and set $h := g^a \pmod p$.
3. Output $\mathsf{pp} = (g, h, p, q)$

$\mathsf{Pedersen.Com}(\mathsf{pp}, m \in \mathbb{Z}_q)$:
1. Sample $r \xleftarrow{\$} \mathbb{Z}_q$.
2. Output $\mathsf{com} = g^r \cdot h^m \pmod p$, and $\mathsf{aux} = (m, r)$.

$\mathsf{Pedersen.Open}(\mathsf{pp}, \mathsf{com}, \mathsf{aux})$:
1. Output $\pi = (m, r)$.

$\mathsf{Pedersen.Verify}(\mathsf{pp}, \mathsf{com}, \pi = (m, r))$:
1. Output 1 if $\mathsf{com} = g^r \cdot h^m$.

Pedersen commitments are perfectly hiding, and computationally binding assuming the hardness of DLog over $\mathbb{Z}_p$.

4 Constraining the Naor-Reingold PRF

In this section, we first describe how to obtain a constrained PRF in the ROM from the Naor-Reingold PRF for the class of inner-product membership constraints, defined below. Then, we detail several optimizations and provide some simple applications for the resulting CPRF.

4.1 Inner Product Membership CPRF from Naor-Reingold

We define the class of inner-product membership (IPM) constraints as $\mathsf{IPM} = \{C^S_{\mathbf{z}} \mid \mathbf{z} \in \mathcal{R}^n, S \subseteq \mathcal{I}\}$, for some sets $\mathcal{R}$ and $\mathcal{I}$, and $n > 0$, where $C^S_{\mathbf{z}} : \mathcal{R}^n \to \{0,1\}$ is defined as $C^S_{\mathbf{z}}(\mathbf{x}) = 0$ iff $\langle \mathbf{z}, \mathbf{x} \rangle \in S$ for an input $\mathbf{x} \in \mathcal{R}^n$. In the following, we first consider binary inputs, i.e., $\mathcal{R} = \{0,1\}$, and later in the section, we explain how we can operate over non-binary inputs, e.g., considering $\mathcal{R} = \{0, 1, \ldots, p-1\}$ for a prime p.

In Figure 6, we describe our construction for constraining the Naor-Reingold PRF for the class of inner-product membership constraints.

Naor-Reingold CPRF for IPM (Binary Inputs)

`Requires:`

- p is a safe prime, i.e., $p = 2q + 1$ for some prime q.
- The input and constraint space is $\{0,1\}^n$.
- The inner-product space is $\mathcal{I} = \{0, 1, \ldots, n\}$.

$\mathsf{CPRF.KeyGen}(1^\lambda)$:

- Run $(\mathbb{G}, g, p) \xleftarrow{\$} \mathsf{GenPar}(1^\lambda)$.
- Sample $\mathbf{a} = (a_0, \ldots, a_n) \xleftarrow{\$} \mathbb{Z}_p^{n+1}$.
- Set and output $\mathsf{msk} = \mathbf{a}$ and $\mathsf{pp} = (\mathbb{G}, g, p)$.

$\mathsf{CPRF.Eval}(\mathsf{pp}, \mathsf{msk}, \mathbf{x} \in \{0,1\}^n)$:

- Parse $\mathsf{pp} = (\mathbb{G}, g, p)$ and $\mathsf{msk} = \mathbf{a}$.
- Output $y = g^{a_0 \cdot \prod_{i=1}^{n} a_i^{x_i}}$.

$\mathsf{CPRF.Constrain}(\mathsf{pp}, \mathsf{msk}, (\mathbf{z}, S))$:

- Parse $\mathsf{pp} = (\mathbb{G}, g, p)$, and $\mathsf{msk} = \mathbf{a}$.
- Sample $r \xleftarrow{\$} \mathbb{Z}_p^*$.
- For $i \in [n]$, set $\alpha_i := a_i \cdot r^{-z_i}$.
- Let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$.
- For $s \in S$, compute $g_s := g^{a_0 \cdot r^s}$.
- Output $\mathsf{ck} = (\boldsymbol{\alpha}, (g_s)_{s \in S}, \mathbf{z})$.

$\mathsf{CPRF.CEval}(\mathsf{pp}, \mathsf{ck}, \mathbf{x} \in \{0,1\}^n)$:

- Parse $\mathsf{pp} = (\mathbb{G}, g, p)$, and $\mathsf{ck} = (\boldsymbol{\alpha}, (g_s)_{s \in S}, \mathbf{z})$.
- Let $s_{\mathbf{x}} := \langle \mathbf{z}, \mathbf{x} \rangle$.
- If $\langle \mathbf{z}, \mathbf{x} \rangle \in S$, output $y = (g_{s_{\mathbf{x}}})^{\prod_{i=1}^{n} \alpha_i^{x_i}}$.
- Otherwise, return $\perp$.

Fig. 6. Naor-Reingold CPRF for IPM constraints over binary inputs.

We first show that, our construction is no-evaluation secure under the sparse power-DDH assumption (Assumption 2).

Theorem 2 (No-Evaluation Security). *Assuming the hardness of sparse power-DDH (Assumption 2), the construction in Figure 6 is a single-key, no-evaluation secure CPRF for the class of* IPM *constraints.*

Proof. We now prove correctness and no-evaluation security of the construction provided in Figure 6.

Correctness. *Correctness follows by inspection after replacing each α_i by $r^{z_i} a_i$ in the output of the* CEval *algorithm.*

No Evaluation Pseudorandomness. *Let $\mathcal{A}$ denote a 1-key no-evaluation adversary against the pseudorandomness of the above construction. Consider the following sequence of hybrid games:*

Hybrid $\mathcal{H}_0$*: This is the standard CPRF security game where the challenge query is answered by returning the output of the CPRF evaluation algorithm. Here the view of the adversary is as follows:*

$$\mathsf{View}_0^{\mathcal{A}} = (\mathsf{pp}, (\mathbf{z}, S), \mathsf{ck}_{(\mathbf{z},S)}, \mathbf{x}^*, y^*),$$

where $\mathsf{pp} = (\mathbb{G}, g, p)$, $\mathsf{ck}_{(\mathbf{z},S)} = (\boldsymbol{\alpha}, (g_s)_{s \in S}, \mathbf{z})$, *and* $y^* = g^{a_0 \cdot \prod_{i=1}^{n} a_i^{x_i^*}}$.

Hybrid $\mathcal{H}_1$*: In this game, the challenger modifies the way it computes the constrained key. It sets the constrained key to be* $\mathsf{ck} = (\boldsymbol{\alpha}, (g_s)_{s \in S}, \mathbf{z})$, *where* $g_s = g^{a_0 \cdot r^s}$ *for each* $s \in S$, *same as in* $\mathcal{H}_0$, *but differently, for* $\boldsymbol{\alpha}$, *it samples a uniform vector* $\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^n$.
Note that in $\mathcal{H}_0$ *we have* $\alpha_i = r^{-z_i} a_i$, *where* a_i *is uniformly sampled from* $\mathbb{Z}_p$ *for each* $i \in [n]$. *Therefore, the distribution of* $\boldsymbol{\alpha}$ *is identical in both games.*

Hybrid $\mathcal{H}_2$*: In this game, the challenger replies to the challenge query by returning* g^t *for a random element* $t \xleftarrow{\$} \mathbb{Z}_p^*$. *We claim that assuming the sparse power-DDH assumption (Assumption 2),* $\mathcal{H}_2$ *and* $\mathcal{H}_1$ *are computationally indistinguishable. Suppose* $\mathcal{A}$ *succeeds in distinguishing these two hybrid games. We construct an adversary* $\mathcal{B}$ *that breaks Assumption 2 with respect to the set* S *and* $\ell = n$.

$\mathcal{B}$ *is given the group* $\mathbb{G} = \langle g \rangle$ *of order* p *and sets* $\mathsf{pp} := (\mathbb{G}, g, p)$ *which it sends to* $\mathcal{A}$. *Then, after receiving a constraint query* $(\mathbf{z}, S)$ *from* $\mathcal{A}$, $\mathcal{B}$ *asks its challenger for a challenge distribution with respect to the set* S *and* $\ell = n$. *It receives a tuple of the form:*

$$\left(g, p, (g^{a \cdot r^s})_{s \in S}, (g^{t_s})_{s \in [n] \setminus S}\right),$$

where $a, r \xleftarrow{\$} \mathbb{Z}_p^*$, *and for each* g^{t_s}, *where* $s \in [n] \setminus S$, *it either holds that* $t_s = a \cdot r^s$ *or* $t_s \xleftarrow{\$} \mathbb{Z}_p^*$.

$\mathcal{B}$ *then selects a random vector* $\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^n$, *sets the constrained key* $\mathsf{ck} = (\boldsymbol{\alpha}, (g^{a \cdot r^s})_{s \in S}, \mathbf{z})$, *and returns it to* $\mathcal{A}$.

To answer the challenge query $\mathbf{x}^*$ *made by* $\mathcal{A}$ *which satisfies* $\langle \mathbf{x}^*, \mathbf{z} \rangle \notin S$*, it selects the* $g^{t_{s^*}}$ *corresponding to* $s^* = \langle \mathbf{x}^*, \mathbf{z} \rangle \pmod p$ *and outputs* $(g^{t_{s^*}})^{\prod_{i=1}^{n} \alpha_i^{x_i^*}}$.

If $t_{s^*} = a \cdot r^{\langle \mathbf{x}^*, \mathbf{z} \rangle}$*, then* $\mathcal{B}$ *simulates the view of* $\mathcal{A}$ *as in Hybrid* $\mathcal{H}_1$*, and otherwise, if* $t_{s^*} \xleftarrow{\$} \mathbb{Z}_p^*$*, it simulates* $\mathcal{H}_2$*. Therefore, distinguishing these two hybrids implies breaking Assumption 2 which proves our claim.*

The rest of the proof proceeds by reversing the sequence of hybrid games while leaving the challenge query answered by a uniformly random value. □

While the construction described in Figure 6 is no-evaluation secure, a simple attack can be mounted as soon as 1 evaluation query is allowed, as we remark below. Fortunately, no-evaluation secure CPRFs can be turned into standard secure CPRFs by known techniques [AMN+18], e.g. in the ROM. We provide more details below.

Remark 2 (A single query attack). Let $(\mathbf{z}, S)$ be the constraint selected by the adversary. For any input $\mathbf{x}$, let us define $s_{\mathbf{x}} = \langle \mathbf{z}, \mathbf{x} \rangle$ and $u_{\mathbf{x}} = \prod_{i=1}^{n} \alpha_i^{x_i}$, where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ is part of the constrained key obtained by the adversary. Then, remark that the output of the CPRF on any input $\mathbf{x}$ can be written as

$$\mathsf{Eval}(\mathbf{x}) = g^{a_0 \cdot r^{s_{\mathbf{x}}} \cdot u_{\mathbf{x}}}$$

where $g^{a_0 \cdot r^{s_{\mathbf{x}}}}$ is part of the constrained key if and only only $s_{\mathbf{x}} \in S$.

Since $u_{\mathbf{x}}$ is computable for any $\mathbf{x}$ from the constrained key, an adversary $\mathcal{A}$ with access to the evaluation oracle, can ask for the output of the CPRF on any input $\mathbf{x}$ such that $\mathbf{s}_{\mathbf{x}} \notin S$ and recover $g^{a_0 \cdot r^{s_{\mathbf{x}}}}$ by raising the evaluation to the power $1/u_{\mathbf{x}}$. Given $g^{a_0 \cdot r^{s_{\mathbf{x}}}}$, $\mathcal{A}$ can now evaluate the CPRF on any input $\mathbf{x}'$ such that $s_{\mathbf{x}} = s_{\mathbf{x}'} \pmod p$, and therefore break the security by finding any input $\mathbf{x}^* \neq \mathbf{x}$ such that $s_{\mathbf{x}} = s_{\mathbf{x}^*} \pmod p$.

Achieving Selective and Adaptive Security. As shown in [AMN+18], our no-evaluation secure CPRF for IPM constraints (Figure 6) can be modified to achieve adaptive security using a hash function modeled as a random oracle. In order to prevent the attack explained above, we can simply hash the output of our no-evaluation secure CPRF. Modeling the hash function as a random oracle, the output of the evaluation function is perfectly random as long as an adversary cannot efficiently find the hash input, i.e., as explain in our attack, cannot find two values $\mathbf{x} \neq \mathbf{x}' \in \mathbb{Z}_p^n$ for which it holds that $s_{\mathbf{x}} = s_{\mathbf{x}'} \notin S$ and $u_{\mathbf{x}} = u_{\mathbf{x}'}$. Since each a_i (therefore each α_i) is a random element of $\mathbb{Z}_p$, the probability that $u_{\mathbf{x}} = u_{\mathbf{x}'}$ for any $\mathbf{x} \neq \mathbf{x}' \in \mathbb{Z}_p^n$ is $1/p$. Therefore, the probability of finding a collision is negligible. The proof of adaptive security proceeds in the same way as in [AMN+18] (Section 4.3). Looking more closely, we can replace the random oracle by a correlation-robust hash function and achieve selective security. Variants of correlation-robust hash have been used in many previous works, see [IKNP03, KKRT16, AMN+18] for a small sample. As in these works, we note that this is a simple standard-model assumption that is likely to hold for classical hash functions such as SHA3.

Non-Binary Inputs. Extending the input domain of our CPRF construction to support non-binary inputs can be simply done by considering the construction presented in Figure 6 for the class of IPM constraints for a larger set $\mathcal{I}$ as the inner-product space. In other words, one can consider $\mathsf{IPM} = \{C_{\mathbf{z}}^S \mid \mathbf{z} \in \mathcal{R}^n, S \subset \mathcal{I}\}$, with $\mathcal{R} = \mathbb{Z}_p$ for a large (safe) prime p and $\mathcal{I} = \{0, 1, \ldots, \ell\}$, with $\ell = \mathsf{poly}(\lambda), \ell \ll p$. The no-evaluation security then follows from Theorem 2 assuming the hardness of sparse power-DDH assumption for the set $S \subset [\ell]$.

4.2 Compressing the keys

We now show how to generate the elements of a master secret key in our Naor-Reingold CPRF by evaluating pseudorandom generators on short seeds. We can thus store a shorter master secret key and communicate a shorter constrained key (for some indices of the constraint vector).

Due to the works of Nechaev [Nec94] and Shoup [Sho97], generic algorithms that solve the DLog problem over $\mathbb{F}_p$, for a prime p, run in $\sqrt{p}$ steps. Therefore, in order to achieve λ bits of security against such algorithms, it should hold that $\log(p) = 2\lambda$. As a result, we can choose the seeds of our PRGs (that generate the master secret key) as short as $\log(p)/2$ bits without any security loss.

Compressing msk. Using a pseudorandom generator (PRG) $\mathsf{G} : \{0,1\}^\lambda \to \{0,1\}^{(n+1)\cdot 2\lambda}$, we can generate the vector $\mathsf{msk} = (a_0, a_1, \ldots, a_n)$ as the output of G on a random λ-bit seed. Doing so, the size of the stored master secret key is reduced from $(n+1)\cdot 2\lambda$ bits to λ bits.

Compressing ck. Recall that for a constraint vector $\mathbf{z}$ and a set S, a constrained key generated in the CPRF construction of Figure 6 is of the form $\mathsf{ck}_{(\mathbf{z},S)} = (\boldsymbol{\alpha}, (g_s)_{s\in S}, \mathbf{z})$, where $\alpha_i = r^{-z_i} a_i$ for all $i \in [n]$. In the case of binary inputs, for each index $i \in [n]$, it holds that $\alpha_i = a_i$, if $z_i = 0$, and $\alpha_i = r^{-1} \cdot a_i$, if $z_i = 1$. In other words, for all the indices $i \in [n]$, where $z_i = 0$, the master secret key element a_i is included in the constrained key and given to the adversary in plain.
Here, we propose an alternative way of generating the master key elements $(a_1, \ldots, a_n) \in \mathbb{Z}_p^n$ which results in including shorter elements in the constrained key and thus reducing the constrained key size. The idea is to use a pseudorandom generator (PRG) $\mathsf{G}' : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ and to generate each a_i as the image of G' on a randomly sampled short seed $\mathsf{seed}_i \xleftarrow{\$} \{0,1\}^\lambda$, for all $i \in [n]$. Doing so, when generating a constrained key for a constraint vector $\mathbf{z}$, for all the indices $i \in [n]$ where $z_i = 0$, we can include the λ-bit seed_i instead of the 2λ-bit master secret key element a_i.

Combining the two above solutions for reducing the msk and ck sizes, we can first use a PRG $\mathsf{G} : \{0,1\}^\lambda \to \{0,1\}^{(2+n)\lambda}$ to generate a vector $(a_0, \mathsf{seed}_1, \ldots, \mathsf{seed}_n)$ from a random λ-bit seed, and afterwards, use another PRG $\mathsf{G}' : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ to generate $a_i \leftarrow \mathsf{G}'(\mathsf{seed}_i)$ for all $i \in [n]$.

Figure 7 presents the modified construction with reduced key size. The steps that are different from the original construction (Figure 6) are marked by the symbol $\trianglerighteq$.

Naor-Reingold CPRF with Compressed Keys

`Requires:`

- p is a safe prime, i.e., $p = 2q + 1$ for some prime q.
- The input and constraint space is $\{0,1\}^n$.
- The inner-product space $\mathcal{I} = \{0, 1, \ldots, n\}$.
- $\mathsf{G} : \{0,1\}^\lambda \to \{0,1\}^{(n+2)\lambda}$, and $\mathsf{G}' : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ are PRGs.

$\mathsf{CPRF.KeyGen}(1^\lambda)$:

- Run $(\mathbb{G}, g, p) \xleftarrow{\$} \mathsf{GenPar}(1^\lambda)$.
- $\trianglerighteq$ Sample $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$.
- $\trianglerighteq$ Output $\mathsf{msk} = \mathsf{seed}$, and $\mathsf{pp} = (\mathbb{G}, g, p)$.

$\mathsf{CPRF.Eval}(\mathsf{pp}, \mathsf{msk}, \mathbf{x} \in \{0,1\}^n)$:

- $\trianglerighteq$ Parse $\mathsf{pp} = (\mathbb{G}, g, p)$, and $\mathsf{msk} = \mathsf{seed}$.
- $\trianglerighteq$ $(a_0, \mathsf{seed}_1, \ldots, \mathsf{seed}_n) \leftarrow \mathsf{G}(\mathsf{seed})$.
- $\trianglerighteq$ For all $i \in [n]$: compute $a_i \leftarrow \mathsf{G}'(\mathsf{seed}_i)$.
- Output $y = g^{a_0 \cdot \prod_{i=1}^{n} a_i^{x_i}}$.

$\mathsf{CPRF.Constrain}(\mathsf{pp}, \mathsf{msk}, (\mathbf{z}, S))$:

- $\trianglerighteq$ Parse $\mathsf{pp} = (\mathbb{G}, g, p)$, and $\mathsf{msk} = \mathsf{seed}$.
- $\trianglerighteq$ $(a_0, (\mathsf{seed}_i)_{i=1}^n) \leftarrow \mathsf{G}(\mathsf{seed})$.
- Sample $r \xleftarrow{\$} \mathbb{Z}_p^*$.
- $\trianglerighteq$ For $i \in [n]$:
  1. If $z_i = 0$, set $\widetilde{\alpha}_i := \mathsf{seed}_i$.
  2. If $z_i = 1$, set $\widetilde{\alpha}_i := r^{-1} \cdot \mathsf{G}'(\mathsf{seed}_i)$.
- $\trianglerighteq$ Let $\widetilde{\alpha} = (\widetilde{\alpha}_1, \ldots, \widetilde{\alpha}_n)$.
- For $s \in S$, set $g_s := g^{a_0 \cdot r^s}$.
- $\trianglerighteq$ Output $\mathsf{ck} = (\widetilde{\alpha}, (g_s)_{s\in S}, \mathbf{z})$.

$\mathsf{CPRF.CEval}(\mathsf{pp}, \mathsf{ck}, \mathbf{x} \in \{0,1\}^n)$:

- $\trianglerighteq$ Parse $\mathsf{pp} = (\mathbb{G}, g, p)$, and $\mathsf{ck} = (\widetilde{\alpha}, (g_s)_{s\in S}, \mathbf{z})$.
- $\trianglerighteq$ For $i \in [n]$:
  1. If $z_i = 0$, set $\alpha_i := \mathsf{G}'(\widetilde{\alpha}_i)$.
  2. If $z_i = 1$, set $\alpha_i := \widetilde{\alpha}_i$.
- Let $s_{\mathbf{x}} := \langle \mathbf{z}, \mathbf{x} \rangle$.
- If $\langle \mathbf{z}, \mathbf{x} \rangle \in S$, output $y = (g_{s_{\mathbf{x}}})^{\prod_{i=1}^{n} \alpha_i^{x_i}}$.
- Otherwise, output $\perp$.

Fig. 7. Naor-Reingold CPRF for IPM with Compressed Keys.

Security Analysis. We observe that computing the elements of the master secret as outputs of a pseudorandom generator on short seeds does not affect the no-evaluation security of the resulting CPRF except for imposing a negligible loss in the security reduction. More precisely, the proof of the no-evaluation security of the CPRF with optimized constrained key size follows from a sequence of hybrid games similar to the proof of Theorem 2 with an adaptation in Hybrid $\mathcal{H}_1$. We break this hybrid into the two following parts:

Hybrid $\mathcal{H}_1^0$: In this game, the challenger modifies the way it computes some elements of the vector $\boldsymbol{\alpha}$ of the constrained key. To generate a constrained key, the challenger first parses $\mathsf{msk} = \mathsf{seed}$ and computes $(a_0, \mathsf{seed}_1, \ldots, \mathsf{seed}_n) \leftarrow \mathsf{G}(\mathsf{seed})$. It then sets $\widetilde{\alpha}_i := \mathsf{seed}_i$, for all $i \in [n]$ such that $z_i = 0$. And differently from $\mathcal{H}_0$, it samples random elements $\widetilde{\alpha}_i \xleftarrow{\$} \mathbb{Z}_p$, for all $i \in [n]$ such that $z_i = 1$. It then sets the constrained key to be $\mathsf{ck} = (\widetilde{\boldsymbol{\alpha}}, (g_s)_{s \in S}, \mathbf{z})$, where $g_s = g^{a_0 \cdot r^s}$, for all $s \in S$.

Note that Hybrid $\mathcal{H}_1^0$ remains computationally indistinguishable from Hybrid $\mathcal{H}_0$. This is because the only difference between the two hybrids is that for all indices $i \in [n]$ where $z_i = 1$, the element $\widetilde{\alpha}_i$ is sampled as a random element in $\mathcal{H}_1$, rather than being computed as $\widetilde{\alpha}_i = r^{-1} \cdot \mathsf{G}'(\mathsf{seed}_i)$ in $\mathcal{H}_0$. Therefore, if a PPT adversary can distinguish between these two hybrids, there must exist an index i such that a random element $\widetilde{\alpha}_i \xleftarrow{\$} \mathbb{Z}_p$ is distinguished from an element of the form $\widetilde{\alpha}_i = r^{-1} \cdot \mathsf{G}'(\mathsf{seed}_i)$. Such an adversary can be leveraged to distinguish between the outputs of the pseudorandom generator G' with random elements of $\mathbb{Z}_p$.

Hybrid $\mathcal{H}_1^1$: In this game, the challenger samples the master secret key without using the PRG G anymore. In other words, the challenger samples $a_0 \xleftarrow{\$} \mathbb{Z}_p$, and $\mathsf{seed}_i \xleftarrow{\$} \{0,1\}^\lambda$ for all $i \in [n]$, and sets $\mathsf{msk} = (a_0, \mathsf{seed}_1, \ldots, \mathsf{seed}_n)$. As a result, when generating a constrained key, it no longer uses G. Here again, Hybrid $\mathcal{H}_1^1$ remains indistinguishable to $\mathcal{H}_1^0$ due to the security of the PRG G.

The rest of the proof is done exactly as in the proof of Theorem 2 with an adaptation that the constrained key element $\boldsymbol{\alpha}$ is now set as in $\mathcal{H}_1^1$.

4.3 Application: A Puncturable PRF in NC^1

A puncturable PRF is a pseudorandom function that allows the generation of a constrained key ck which enables one to evaluate the output of the PRF on all inputs but one. The well-known construction of [GGM84a] from one-way functions offers a puncturable PRF that is computable by a linear-depth circuit in the size of the input. Noticing that the IPM class contains puncturing constraints (for certain parameters), our CPRF construction (Figure 6) which is essentially constrained Naor-Reingold PRF and can be evaluated by a log-depth circuit, offers a puncturable PRF in NC^1.

Construction. For this construction, we consider the input space to be $\{-1,1\}^n$. Suppose that we want to puncture the Naor-Reingold PRF on an input $\mathbf{x}^* \in \{-1,1\}^n$. In what follows, we show that this can be done using the Naor-Reingold CPRF construction of Figure 6 that supports the class of IPM constraints. Note that for a vector $\mathbf{x}^* \in \{-1,1\}^n$ and any vector $\mathbf{x} \in \{-1,1\}^n$, the inner-product $\langle \mathbf{x}, \mathbf{x}^* \rangle = n$ iff $\mathbf{x} = \mathbf{x}^*$. Also, all possible values of the inner-product between vectors in $\{-1,1\}^n$ lie in the set $\mathcal{I} = \{-n, -n+2, -n+4, \ldots, n-2, n\}$. Therefore, setting the constraint set $S = \mathcal{I} \setminus \{n\}$, a CPRF supporting inner-product membership constraints for the set S, can be viewed as a puncturable PRF for any vector $\mathbf{x}^* \in \{-1,1\}^n$. Parameters of the IPM constraints for this application is presented in Figure 8.

Naor-Reingold Puncturable PRF

Setting the parameters of $\mathsf{IPM} = \{C_{\mathbf{z}}^S \mid \mathbf{z} \in \mathcal{R}^n, S \subset \mathcal{I}\}$**:**

- Input and constraint vectors space: $\mathcal{R} = \{-1,1\}^n$.

– Inner-product space: $\mathcal{I} = \{-n, -n+2, -n+4, \ldots, n-2, n\}$.
– Inner-product constraint set: $S = \mathcal{I} \setminus \{n\}$.

Fig. 8. Parameters of IPM for puncturing constraints.

Security. The 1-key selective no-evaluation security of the scheme is implied by the sparse power-DDH assumption. Importantly, we note that in the case of puncturing constraints, the 1-key selective no-evaluation security becomes equivalent to the standard selective security notion where an adversary is allowed to query the evaluation oracle. This is because the challenge query can only be issued on the punctured point $\mathbf{x}^*$. Therefore, in the selective setting, the challenger can compute the constrained key $\mathsf{ck}_{\mathbf{x}^*}$ in the beginning of the security game, and consequently, can answer evaluation queries for inputs $\mathbf{x} \neq \mathbf{x}^*$ by computing and returning the output of $\mathsf{CEval}(\mathsf{pp}, \mathsf{ck}_{\mathbf{x}^*}, \mathbf{x})$ to the adversary. In the following, we briefly go over the security proof:

Hybrid $\mathcal{H}_0$: This is the 1-key selective CPRF security game where the evaluation and challenge queries are answered by returning the output of the CPRF evaluation algorithm on the queried inputs. The view of the adversary in this game is as follows:

$$\mathsf{View}_0^{\mathcal{A}} = (\mathsf{pp}, \{\mathbf{x}_i, y_i\}_{i\in[Q]}, \mathsf{ck}, \mathbf{x}^*, y^*),$$

where $\mathsf{pp} = (\mathbb{G}, g, p)$, $\mathsf{ck} = (\boldsymbol{\alpha}, (g_s)_{s\in S})$, and $y = g^{a_0 \cdot \prod_{i=1}^{n} a_i^{x}}$ for all $(x, y) \in \{(x_i, y_i)_{i\in[Q]}, (\mathbf{x}^*, y^*)\}$.

Hybrid $\mathcal{H}_1$: In this game, we change how the evaluation oracle queries are answered. As we consider the selective setting, the challenger knows the punctured point $\mathbf{x}^*$ from the beginning. It can therefore compute the constrained key ck from the start, and answer an evaluation query on any input $\mathbf{x} \neq \mathbf{x}^*$ by computing $y \leftarrow \mathsf{CEval}(\mathsf{pp}, \mathsf{ck}, \mathbf{x})$. The view of the adversary remains identical to its view in $\mathcal{H}_0$ by the correctness of the CPRF.

Hybrid $\mathcal{H}_2$: In this game, the challenger sets the vector $\boldsymbol{\alpha}$ in the constrained key ck to be a random vector from $\mathbb{Z}_p^*$. Hybrids $\mathcal{H}_1$ and $\mathcal{H}_2$ remain statistically indistinguishable.

Hybrid $\mathcal{H}_3$: In this hybrid, the challenger replies to the challenge query by returning g^t for a random element $t \xleftarrow{\$} \mathbb{Z}_p^*$. Assuming the sparse power-DDH assumption (Assumption 2), hybrids $\mathcal{H}_2$ and $\mathcal{H}_3$ are computationally indistinguishable. Let $\mathcal{A}$ be an adversary that distinguishes $\mathcal{H}_2$ and $\mathcal{H}_3$. In what follows, we construct an adversary $\mathcal{B}$ that breaks Assumption 2 with respect to the set $S' = \{0, 2, 4, \ldots, 2n-2\}$ and $\ell = 2n$.

Knowing the group $\mathbb{G} = \langle g \rangle$ of order p, $\mathcal{B}$ sets $\mathsf{pp} := (\mathbb{G}, g, p)$ and sends it to $\mathcal{A}$.

Then, after receiving a punctured point $\mathbf{x}^*$ from $\mathcal{A}$, adversary $\mathcal{B}$ asks the assumption oracle for a challenge distribution with respect to the set $S' = \{0, 2, 4, \ldots, 2n-2\}$ and $\ell = 2n$ and receives a tuple of the following form:

$$\left(g, p, (g_s)_{s\in S'}, (g_s)_{s\in[2n]\setminus S'}\right),$$

where for all $s \in S'$ it holds that $g_s = g^{a\cdot r^s}$ for some $a, r \xleftarrow{\$} \mathbb{Z}_p^*$, and for all $s \in [2n] \setminus S'$ it either holds that $g_s = g^{a\cdot r^s}$ or $g_s = g^{t_s}$ for some $t_s \xleftarrow{\$} \mathbb{Z}_p^*$.

$\mathcal{B}$ then selects a random vector $\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^n$ and sets the constrained key $\mathsf{ck} = (\boldsymbol{\alpha}, (g^{a\cdot r^s})_{s\in S'}, \mathbf{z})$. Note that with overwhelming probability, a can be written as $a = r^{-n} \cdot \beta$, for a random element $\beta \in \mathbb{Z}_p^*$. Therefore we can rewrite $g^{a\cdot r^s} = g^{\beta \cdot r^{s-n}}$ for all $s \in S'$. As a result, the tuple $(g^{a\cdot r^s})_{s\in S'}$ for the set $S' = \{0, 2, 4, \ldots, 2n-2\}$ simulates $(g^{\beta\cdot r^s})_{s\in S}$, where $S = \{-n, -n+2, \ldots, n-4, n-2\}$.

Finally, to answer the challenge query on the punctured input $\mathbf{x}^*$ which satisfies $\langle \mathbf{x}^*, \mathbf{x}^* \rangle = n$, $\mathcal{B}$ selects the g_{2n} and outputs $y^* = (g_{2n})^{\prod_{i=1}^{n} \alpha_i^{x_i^*}}$. If $\mathcal{A}$ can distinguish between hybrids $\mathcal{H}_3$ and $\mathcal{H}_2$, the adversary $\mathcal{B}$ can successfully break the sparse power-DDH assumption with respect to set S' and $\ell = 2n$. □

5 Fast PCFs for OTs from Pseudorandomly Constrained PRFs

In this section, we provide a general framework for building programmable PCFs for OT correlations from CPRFs and show how it can be instantiated under standard assumptions, yielding a concretely efficient construction.

In section 5.1 we provide a general framework for building programmable PCFs from CPRFs supporting classes of "pseudorandom constraints". In section 5.2 we introduce a more specific template, which uses an "inner product membership (weak) PRF" to ensure these pseudorandom constraints can be expressed as "inner-product membership" constraints (which is the class tolerated by the Naor-Reingold CPRF section 4). In section 5.3 we show that many weak PRFs based on LWR or random CSPs fit our framework of inner product membership (weak) PRF. Finally, we show in section 5.4 that, when instantiated with the Naor-Reingold CPRF, our PCF natively admits a 2-round low-communication protocol for securely distributing the PCF key generation.

5.1 General Template

In this section, we provide constructions of (weak/strong) PCF for OT correlations from various notions of CPRF supporting a "hardcoded-key (weakly/strongly) pseudorandom class of constraints". While technically incomparable, our constructions can be seen as achieving increasingly "stronger" PCFs from increasively "stronger" assumptions on the CPRF.

1. We first show in section 5.1 how to obtain a [weak/strong] PCF for Rabin-OT correlations, given a CPRF supporting as constraint a [weak/strong] PRF family whose key is hardcoded.
2. We then show in fig. 9 how to obtain a (weak/strong) PCF for random-OT correlations, given a CPRF supporting as constraint a (weak/strong) PRF family whose key is hardcoded as well as its opposite.
   A minor drawback of this construction as opposed to the previous one is that the PCF key is roughly twice as large (as it now contains two CPRF keys), but it achieves the more standard notion of "PCF for OT correlation", without having to use the relatively costly generic transformation from Rabin-OT to random-OT. The assumption is strictly stronger however, as a CPRF for $\mathcal{C}$ may not necessarily imply the existence of a CPRF for $\mathcal{C} \cup \{1 - f : f \in \mathcal{C}\}$.
3. Finally—and as our main construction of this section—we show in section 10 how to obtain a PCF for random-OT correlations, given a CPRF supporting the class of "1-in-2 pseudorandom constraints". That is, for every $x \in \{0,1\}^{n-1}$ (where n is the input length), exactly one of $x|0$ or $x|1$ is constrained, and which one is pseudorandom (with respect to one hardcoded key). A formal definition is provided in definition 12.
   The main difference with the previous construction is that the PCF key now contains a single CPRF key, at the cost of assuming a PCF for a more involved class of constraints. This is greatly mitigated by the fact that several low-complexity weak PRF candidates from the literature admit a low-complexity circuit for "1-in-2 pseudorandom selection".

PCF for Rabin-OT from CPRF supporting wPRF Constraints. We start by showing how one can build a PCF for Rabin-OT correlations using a CPRF supporting (weak) PRF as constraint.

Definition 11 (Oblivous Transfer Correlations). *We consider two flavours of Oblivious Transfer correlations (OT correlations).*

- *The* Rabin OT correlation over message space $\mathcal{M}$ *can be defined as being sampled as a pair* m *and* m'*, where* $m \xleftarrow{\$} \mathcal{M}$ *is the OT sender's random message and* $m' \xleftarrow{\$} \{m, \perp\}$ *is the message given to the receiver, which is equal to* m *with probability* $1/2$ *and to* $\perp$ *(a special symbol not in* $\mathcal{M}$*, signifying the receiver does not get the sender message) with probability* $1/2$*.*
- *The* (random, 1-out-of-2) OT correlation over message space $\mathcal{M}$ *can be defined as being sampled as a pair (of pairs)* (m_0, m_1) *and* (σ, m_σ)*, where* $(m_0, m_1) \xleftarrow{\$} \mathcal{M}^2$ *is the OT sender's pair of random messages and* $\sigma \xleftarrow{\$} \{0,1\}$ *is the random selection bit given to the receiver.*

Crépeau [Cré88] showed that the two flavours of OT considered in definition 11 are essentially equivalent. However, the transformation from Rabin-OT to 1-out-of-2 OT is interactive[12], and expends λ Rabin-OTs in order to perform a single 1-out-of-2 OT (which is the notion standardly used in MPC). Since the construction of fig. 9 is a PCF for Rabin-OT correlations, we view it as more of an introductory "proof of concept".

PCF Pre-Computable weak/strong PCF for Rabin-OT Correlations (with a single call to a CPRF)

`Requires:`

- $F\colon \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \{0,1\}$ is a weak/strong PRF.
- $\mathsf{CPRF} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ is a constrained PRF supporting as constraints $\{F(k,\cdot)\colon k \in \mathcal{K}_\lambda\}$.

$\mathsf{PCF.Gen}_0(1^\lambda)$:

1. Run $(\mathsf{pp}, \mathsf{msk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$.
2. Set $k_0 \leftarrow (\mathsf{pp}, \mathsf{msk})$.
3. Output k_0.

$\mathsf{PCF.Gen}_1(1^\lambda, k_0)$:

1. Sample $k \xleftarrow{\$} \mathcal{K}_\lambda$.
2. Parse k_0 as $k_0 = (\mathsf{pp}, \mathsf{msk})$.
3. Run $\mathsf{ck} \leftarrow \mathsf{Constrain}(\mathsf{msk}, F(k,\cdot))$.
4. Set $k_1 \leftarrow (\mathsf{ck}, k, \mathsf{pp})$.
5. Output k_1.

$\mathsf{PCF.Eval}(1^\lambda, \sigma, k_\sigma, x)$:

1. If $\sigma = 0$:
   (a) Parse k_0 as $k_0 = (\mathsf{pp}, \mathsf{msk})$.
   (b) Set $y_0 \leftarrow \mathsf{Eval}(\mathsf{pp}, \mathsf{msk}, x)$.
   (c) Output y_0.

2. If $\sigma = 1$:
   (a) Parse k_1 as $k_1 = (\mathsf{ck}, k, \mathsf{pp})$.
   (b) Compute $b \coloneqq F(k, x)$.
   (c) If $b = 0$, $y_1 \leftarrow \mathsf{CEval}(\mathsf{pp}, \mathsf{ck}, x)$.
   (d) If $b = 1$, $y_1 \leftarrow \perp$.
   (e) Output y_1.

Fig. 9. Pre-Computable wPCF/PCF for Rabin-OT from CPRF supporting (w)PRF constraints.

Theorem 3 ((w/s)PCF for Rabin-OT from CPRF supporting (w/s)PRF Constraints). *Assume the existence of the following primitives:*

- $F\colon \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \{0,1\}$ *is a weak (resp. strong) PRF.*
- $\mathsf{CPRF} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ *is a secure CPRF supporting as constraints the class* $\{F(k,\cdot)\colon k \in \mathcal{K}_\lambda\}$.

Then the construction of fig. 10 is a secure weak (resp. strong) precomputable PCF for Rabin-OT correlations.

Because our main goal is building PCFs for 1-in-2 OT correlations (and the above is little more than a toy example), we forgo a full proof in this version of the paper and instead sketch the sequence of hybrids which can be used to formally prove theorem 3.

- *Weakly (resp. Strongly) Pseudorandom Rabin-OT Correlated Outputs.* By correctness of CPRF, whenever $F(k,x) = 0$ it holds that with all but negligible probability $\mathsf{Eval}(\mathsf{pp}, \mathsf{msk}, x) = \mathsf{CEval}(\mathsf{pp}, \mathsf{ck}, x)$ if ck is a key constrained by $F(k,\cdot)$. It follows that in the experiment $\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},N,0}(\lambda)$, y_1 can be generated as follows while only introducing a negligible security loss:

$$y_1 \leftarrow \begin{cases} y_0 & \text{If } b = 0 \\ \perp & \text{Else} \end{cases}.$$

[12] Crépeau's transformation does not seem to yield a procedure allowing parties holding Rabin-OT-correlated shares to generate 1-out-of-2-OT-correlated shares locally.

The next hybrid involves generating y_0 as $y_0 \xleftarrow{\$} \mathcal{M}$ (instead of $y_0 \leftarrow \mathsf{Eval}(\mathsf{pp}, \mathsf{msk}, x)$). By CPRF security, the outputs of the master evaluation algorithm of a CPRF are pseudorandom from the point of view of an external adversary without knowledge of the master key (or any constrained key). It follows that this hybrid only introduces a negligible security loss. The final hybrid consists in sampling $b \xleftarrow{\$} \{0,1\}$ instead of $b \leftarrow F(k,x)$. This step incurs a negligible security loss by (weak/strong) PRF security, as the outputs of a PRF are pseudorandom from the point of view of an external adversary.

– *Weak (Resp. Strong) PCF Security.* Security for $\sigma = 0$ (*i.e.* "secrecy of y_1 given k_0") follows from the fact that k_0 contains no information about the PRF key k, and hence an internal adversary with knowledge of k_0 has only negligeable advantage in determining on which values of x $\mathsf{Eval}(1, k_1, x) = \bot$. Security for $\sigma = 1$ follows from CPRF security: the constrained key ck in k_1 does not allow for evaluation of points x such that $F(k,x) = 1$.

The outputs of the master evaluation algorithm of a CPRF, as well as the outputs of a weak/strong PRF on random/arbitrary inputs, are pseudorandom from the point of view of an external adversary without knowledge of the keys. Furthermore,

PCF for OT from CPRF supporting wPRF Constraints and their Opposite. We now provide a construction of a PCF for 1-out-of-2 random OT correlations (which is the standard notion of "PCF for OT correlations"), which does not rely on the generic transformation from Rabin-OT. Our construction relies on a CPRF which can be constrained with respect to both a (weak) PRF and its opposite. More specifically, for a (weak) PRF F with range $\{0,1\}$, we require a CPRF supporting the class of constraints $\{F(k,\cdot) \mid k \in \mathcal{K}_\lambda\}$ as well as a CPRF supporting the class of constraints $\{1 - F(k,\cdot) \mid k \in \mathcal{K}_\lambda\}$. However the existence of CPRFs supporting X and Y generically implies the existence of a CPRF supporting $X \cup Y$.

PCF Pre-Computable weak/strong PCF for OT Correlations (with two calls to a CPRF)

`Requires:`

– $F\colon \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \{0,1\}$ is a weak/strong PRF; we denote $\overline{F}\colon \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \{0,1\}$ the function defined by $\overline{F}(k,x) \coloneqq 1 - F(k,x)$.
– $\mathsf{CPRF} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ is a constrained PRF supporting as constraints $\{F(k,\cdot)\colon k \in \mathcal{K}_\lambda\} \cup \{\overline{F}(k,\cdot)\colon k \in \mathcal{K}_\lambda\}$.

$\mathsf{PCF.Gen}_0(1^\lambda)$:

1. Run $(\mathsf{pp}_0, \mathsf{msk}_0) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$.
2. Run $(\mathsf{pp}_1, \mathsf{msk}_1) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$.
3. Set $k_0 \leftarrow (\mathsf{pp}_0, \mathsf{pp}_1, \mathsf{msk}_0, \mathsf{msk}_1)$.
4. Output k_0.

$\mathsf{PCF.Gen}_1(1^\lambda, k_0)$:

1. Sample $k \xleftarrow{\$} \mathcal{K}_\lambda$.
2. Parse k_0 as $k_0 = (\mathsf{pp}_0, \mathsf{pp}_1, \mathsf{msk}_0, \mathsf{msk}_1)$.
3. Run $\mathsf{ck}_0 \leftarrow \mathsf{Constrain}(\mathsf{msk}_0, F(k,\cdot))$.
   // For inputs x such that $F(k,x) = 0$.
4. Run $\mathsf{ck}_1 \leftarrow \mathsf{Constrain}(\mathsf{msk}_1, \overline{F}(k,\cdot))$.
   // For inputs x such that $F(k,x) = 1$.
5. Set $k_1 \leftarrow (\mathsf{ck}_0, \mathsf{ck}_1, k, \mathsf{pp}_0, \mathsf{pp}_1)$.
6. Output k_1.

$\mathsf{PCF.Eval}(1^\lambda, \sigma, k_\sigma, x)$:

1. If $\sigma = 0$:
   (a) Parse k_0 as $k_0 = (\mathsf{pp}_0, \mathsf{pp}_1, \mathsf{msk}_0, \mathsf{msk}_1)$.
   (b) Set $r_0 \leftarrow \mathsf{Eval}(\mathsf{pp}_0, \mathsf{msk}_0, x)$.
   (c) Set $r_1 \leftarrow \mathsf{Eval}(\mathsf{pp}_1, \mathsf{msk}_1, x)$.
   (d) Set $y_0 \leftarrow (r_0, r_1)$.
   (e) Output y_0.
2. If $\sigma = 1$:
   (a) Parse k_1 as $k_1 = (\mathsf{ck}_0, \mathsf{ck}_1, k, \mathsf{pp}_0, \mathsf{pp}_1)$.
   (b) Compute $b \coloneqq F(k,x)$.
   (c) Compute $r \leftarrow \mathsf{CEval}(\mathsf{pp}_b, \mathsf{ck}_b, x)$.
   (d) Set $y_1 \leftarrow (b, r)$.
   (e) Output y_1.

Fig. 10. Pre-Computable wPCF/PCF for OT from CPRF supporting (w)PRF constraints and their opposite.

Theorem 4 (PCF from CPRF supporting (w)PRF constraints and their opposite). *Assume the existence of the following primitives:*

- $F\colon \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \{0,1\}$ *is a* (Q,T,ϵ)*-secure weak (resp. strong) PRF; we denote* $\overline{F}\colon \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \{0,1\}$ *the function defined by* $\forall k \in \mathcal{K}_\lambda, \forall x \in \mathcal{X}_\lambda, \overline{F}(k,x) := 1 - F(k,x)$*;*
- $\mathsf{CPRF} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ *is a* (Q',T',ϵ')*-secure* δ*-correct CPRF with output space* $\mathcal{M}$ *supporting the class of constraints* $\{F(k,\cdot)\colon k \in \mathcal{K}_\lambda\} \cup \{\overline{F}(k,\cdot)\colon k \in \mathcal{K}_\lambda\} \cup \{\mathbf{1}\}$ *(*$\mathbf{1}$ *denotes here the constant circuit rejecting all points). Let* $c \in \mathbb{N}$ *be a constant such that* KeyGen*,* Eval*,* Constrain*, and* CEval *all run in time* n^c*.*

Then the construction of fig. 10 is a $(\min\{Q, 2Q'\}, \min\{T - 2n^c, T' - Q''n^c\}, \epsilon + 2\epsilon' + \min\{Q, 2Q'\} \cdot \delta)$*-secure weak (resp. strong) precomputable PCF for OT correlations. The message space of the OT correlations is the output space of* CPRF*.*

Proof. We deal with the case where F is only assumed to be a weak PRF (and consequently we only need to prove that Figure 10 is a weak PCF), but the case where F is a strong PRF is completely analogous.
Let $(Q'', T'', \epsilon'') := (\min\{Q, 2Q'\}, \min\{T, T'\}, \epsilon + 2\epsilon' + \min\{Q, 2Q'\} \cdot \delta)$*. Since the construction of Figure 10 provides* $\mathsf{PCF.Gen}_0$ *and* $\mathsf{PCF.Gen}_1$ *explicitly, precomputability follows from definition. It therefore suffices to show that* $(\mathsf{PCF.Gen}, \mathsf{PCF.Eval})$ *is a* (T'', Q'', ϵ'')*-secure PCF, where* PCF.Gen *is defined as:*

$$\mathsf{PCF.Gen}(1^\lambda): \ (k_0, \mathsf{aux}) \xleftarrow{\$} \mathsf{PCF.Gen}_0(1^\lambda); \ k_1 \xleftarrow{\$} \mathsf{PCF.Gen}_1(1^\lambda, k_0, \mathsf{aux}); \ \textit{Output } (k_0, k_1) \ .$$

- Weakly Pseudorandom OT-Correlated Outputs. *Let* $\mathcal{A}$ *be an adversary that runs in time* T''*. We prove that*

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

where:

$\underline{\mathsf{Exp}^{\mathsf{w\text{-}pr}}_{\mathcal{A},Q'',0}(\lambda)}$:

For $i = 1, \ldots, Q''(\lambda)$:
$x^{(i)} \xleftarrow{\$} \mathcal{X}_\lambda$
$b^{(i)}, r_0^{(i)}, r_1^{(i)} \xleftarrow{\$} \{0,1\}$
$y_0^{(i)} \leftarrow (r_0^{(i)}, r_1^{(i)})$; $y_1^{(i)} \leftarrow (b^{(i)}, r_{b^{(i)}}^{(i)})$
$b \leftarrow \mathcal{A}(1^\lambda, (x^{(i)}, y_0^{(i)}, y_1^{(i)})_{i \in [Q''(\lambda)]})$
Output b

$\underline{\mathsf{Exp}^{\mathsf{w\text{-}pr}}_{\mathcal{A},Q'',1}(\lambda)}$:

$(k_0, k_1) \leftarrow \mathsf{PCF.Gen}(1^\lambda)$
For $i = 1, \ldots, Q''(\lambda)$:
$x^{(i)} \xleftarrow{\$} \mathcal{X}_\lambda$
for $\sigma \in \{0,1\}$, $y_\sigma^{(i)} \leftarrow \mathsf{PCF.Eval}(\sigma, k_\sigma, x^{(i)})$
$b \leftarrow \mathcal{A}(1^\lambda, (x^{(i)}, y_0^{(i)}, y_1^{(i)})_{i \in [Q''(\lambda)]})$
Output b

Consider the following sequence of hybrids:

- *Hybrid 1: This is the real-world experiment* $\mathsf{Exp}^{\mathsf{w\text{-}pr}}_{\mathcal{A},Q'',1}(\lambda)$*.*
- *Hybrid 2: For all* $i \in [Q''(\lambda)]$*, replace "*$y_1^{(i)} \leftarrow \mathsf{PCF.Eval}(1, k_1, x^{(i)})$*" with:*
  1. *Parse* $k_1 = (\mathsf{ck}_0, \mathsf{ck}_1, k, \mathsf{pp}_0, \mathsf{pp}_1)$
  2. *Compute* $b^{(i)} := F(k, x^{(i)})$
  3. *Set* $r^{(i)} \leftarrow y_0^{(i)}[b^{(i)}]$
     (i.e. $r^{(i)} \leftarrow y_0^{(i)}$.*first* *if* $b^{(i)} = 0$*, and* $r^{(i)} \leftarrow y_0^{(i)}$.*second* *if* $b^{(i)} = 1$)
  4. *Set* $y_1^{(i)} \leftarrow (b^{(i)}, r^{(i)})$
  5. *Output* $y_1^{(i)}$

  Observe that the only difference between hybrids 1 and 2 is whether $y_1^{(i)}.\mathsf{second}$ *are generated using* CPRF.CEval *(in hybrid 1) or* CPRF.Eval *(in hybrid 2). For* $i \in [Q'']$*, let* E_i *denote the event "*$y_0^{(i)}[b^{(i)}] = r^{(i)}$*" (where the random variables* $y_0^{(i)}$*,* $b^{(i)}$*, and* $r^{(i)}$ *are those from hybrid 1). By the* δ*-correctness of* CPRF*,* $\forall_i \in [Q''], \Pr[E_i] \geq 1 - \delta$*, and further consider the event* $E := \cap_{i \in [Q'']} E_i$*. Observe that* $\Pr[\mathsf{Hyb}_1 = 1 | E] = \Pr[\mathsf{Hyb}_2 = 1]$*. Finally, by the law of total*

probability:

$$\begin{aligned}|\Pr[\mathsf{Hyb}_1=1]-\Pr[\mathsf{Hyb}_2=1]| &= |\Pr[\mathsf{Hyb}_1=1|E]\cdot\Pr[E]+|\Pr[\mathsf{Hyb}_1=1|\overline{E}]\cdot\Pr[\overline{E}]\\ &\qquad -\Pr[\mathsf{Hyb}_2=1]|\\ &= |\Pr[\mathsf{Hyb}_1=1|\overline{E}]\cdot\Pr[\overline{E}]-\Pr[\mathsf{Hyb}_2=1]\cdot(1-\Pr[E])|\\ &= |\Pr[\mathsf{Hyb}_1=1|\overline{E}]-\Pr[\mathsf{Hyb}_2=1]|\cdot(1-\Pr[E])\\ &\le (1-\Pr[E])\le Q''\delta\ .\end{aligned} \tag{1}$$

- *Hybrid 3: For all $i\in[Q'']$, replace $y_0^i\leftarrow\mathsf{PCF.Eval}(0,k_0,x^{(i)})$ with:*
  1. *Parse $k_0=(\mathsf{pp}_0,\mathsf{pp}_1,\mathsf{msk}_0,\mathsf{msk}_1)$*
  2. *Compute $b^{(i)}:=F(k,x^{(i)})$*
  3. *Sample $r_{1-b^{(i)}}^{(i)}\xleftarrow{\$}\mathcal{M}$*
  4. *Set $r_{b^{(i)}}^{(i)}\leftarrow\mathsf{Eval}(\mathsf{pp}_{b^{(i)}},\mathsf{msk}_{b^{(i)}},x)$*
  5. *Set $y_0\leftarrow(r_0,r_1)$*
  6. *Output y_0*

  By the security of CPRF*, evaluations of the master key $\mathsf{msk}_{1-b^{(i)}}$ on constrained points are pseudorandom from the point of view of an adversary given a constrained key. It follows that $|\Pr[\mathsf{Hyb}_3=1]-\Pr[\mathsf{Hyb}_4=1]|\le\epsilon'$.*
- *Hybrid 4: For all $i\in[Q'']$, replace $r_{b^{(i)}}\leftarrow\mathsf{Eval}(\mathsf{pp}_{b^{(i)}},\mathsf{msk}_{b^{(i)}},x)$ with $r_{b^{(i)}}^{(i)}\xleftarrow{\$}\mathcal{M}$.*
  Because $\mathbf{1}$ is in the constraint class, by CPRF security, the outputs of CPRF.Eval *must be pseudorandom from the point of view of an external adversary who is not given any CPRF constrained key. It follows that $|\Pr[\mathsf{Hyb}_3=1]-\Pr[\mathsf{Hyb}_4=1]|\le\epsilon'$.*
- *Hybrid 5: For all $i\in[Q'']$, replace $b^{(i)}:=F(k,x^{(i)})$ with $b^{(i)}\xleftarrow{\$}\{0,1\}$. Consider the following adversary $\mathcal{B}$, whose goal is to attack the PRF F. $\mathcal{B}((x_i,b_i)_{i\in[Q'']})$:*
  1. *$(k_0,k_1)\xleftarrow{\$}\mathsf{PCF.Gen}_0$ (where* PCF *is the construction of fig. 10)*
  2. *For $i\in[Q'']$, Sample $r_0^{(i)},r_1^{(i)}\leftarrow\{0,1\}$*
  3. *Output $\mathcal{A}(1^\lambda,(x^{(i)},r_0^{(i)},r_1^{(i)})_{i\in[Q'']})$*

  Because $\mathcal{A}$ runs in time $T''\le T-2n^c$, $\mathcal{B}$ runs in time at most T and therefore by (Q,T,ϵ)-security if the wPRF F:

$$\left|\Pr\left[\mathcal{B}((x_i,b_i)_{i\in[Q'']})=1\colon\begin{array}{l}x_i\xleftarrow{\$}\mathcal{X}_\lambda\\ b_i\xleftarrow{\$}\{0,1\}\end{array}\right]-\Pr\left[\mathcal{B}((x_i,b_i)_{i\in[Q'']})=1\colon\begin{array}{l}k\xleftarrow{\$}\mathcal{K}_\lambda\\ x_i\xleftarrow{\$}\mathcal{X}_\lambda\\ b_i\leftarrow F(k,x_i)\end{array}\right]\right|\le\epsilon\ .$$

  This is exactly saying that $|\Pr[\mathsf{Hyb}_4=1]-\Pr[\mathsf{Hyb}_5=1]|\le\epsilon$.
- *Hybrid 6: This is the ideal-world experiment $\mathsf{Exp}^{\mathsf{w\text{-}pr}}_{\mathcal{A},Q'',0}(\lambda)$. Observe that hybrids 5 and 6 are code-equivalent.*

By combining all of the above hybrids $|\Pr[\mathsf{Exp}^{\mathsf{w\text{-}pr}}_{\mathcal{A},Q'',0}(\lambda)=1]-\Pr[\mathsf{Exp}^{\mathsf{w\text{-}pr}}_{\mathcal{A},Q'',1}(\lambda)=1]|\le\epsilon''$.

– Weak PCF Security: *Let $\mathcal{A}$ be a time-T'' adversary. We need to show that $|\Pr[\mathsf{Exp}^{\mathsf{w\text{-}sec}}_{\mathcal{A},Q'',\sigma,0}(\lambda)=1]-\Pr[\mathsf{Exp}^{\mathsf{w\text{-}sec}}_{\mathcal{A},Q'',\sigma,1}(\lambda)=1]|\le\epsilon''$, where:*

$\underline{\mathsf{Exp}^{\mathsf{w\text{-}sec}}_{\mathcal{A},Q'',\sigma,0}(\lambda)}$:

$(k_0,k_1)\leftarrow\mathsf{PCF.Gen}(1^\lambda)$
For $i=1,\dots,N(\lambda)$:
  $x^{(i)}\xleftarrow{\$}\{0,1\}^{n(\lambda)}$
  $y_\sigma^{(i)}\xleftarrow{\$}\mathsf{wPCF.Eval}(\sigma,k_\sigma,x^{(i)})$
  $y_{1-\sigma}^{(i)}\xleftarrow{\$}\mathsf{RSample}(1^\lambda,\sigma,y_\sigma^{(i)})$
$*$
$b\leftarrow\mathcal{A}(1^\lambda,\sigma,k_\sigma,(x^{(i)},y_{1-\sigma}^{(i)})_{i\in[N(\lambda)]})$
Output b

$\underline{\mathsf{Exp}^{\mathsf{w\text{-}sec}}_{\mathcal{A},Q'',\sigma,1}(\lambda)}$:

$(k_0,k_1)\leftarrow\mathsf{PCF.Gen}(1^\lambda)$
For $i=1,\dots,Q''$:
  $x^{(i)}\xleftarrow{\$}\{0,1\}^{n(\lambda)}$
  $y_{1-\sigma}^{(i)}\xleftarrow{\$}\mathsf{wPCF.Eval}(1-\sigma,k_{1-\sigma},x^{(i)})$
$b\leftarrow\mathcal{A}(1^\lambda,\sigma,k_\sigma,(x^{(i)},y_{1-\sigma}^{(i)})_{i\in[Q'']})$
Output b

Recall that for OT correlations (with message space $\mathcal{M}$), RSample *is defined as follows.*
$\mathsf{RSample}(1^\lambda,\sigma,y_\sigma)$:

- *If $\sigma = 0$:*
  - *Sample $b \xleftarrow{\$} \{0,1\}$, $r \xleftarrow{\$} \mathcal{M}$*
  - *Output $y_{1-\sigma} = (b, r)$*
- *If $\sigma = 1$:*
  - *Parse y_σ as $y_\sigma = (b, r_b)$*
  - *Sample $r_{1-b} \xleftarrow{\$} \mathcal{M}$*
  - *Output $y_{1-\sigma} = (r_0, r_1)$*

- *If $\sigma = 0$: The only difference between $\mathsf{Exp}^{\mathsf{w\text{-}sec}}_{\mathcal{A},Q'',0,0}(\lambda)$ and $\mathsf{Exp}^{\mathsf{w\text{-}sec}}_{\mathcal{A},Q'',0,1}(\lambda)$ is whether the $(y_1^{(i)}.\mathsf{first})_{i\in[Q'']}$ are sampled truly at random or using a PRF (whose key is unknown to $\mathcal{A}$). By security of the PRF F, we immediately get $|\Pr[\mathsf{Exp}^{\mathsf{w\text{-}sec}}_{\mathcal{A},Q'',0,0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{w\text{-}sec}}_{\mathcal{A},Q'',0,1}(\lambda) = 1]| \leq \epsilon$.*
- *If $\sigma = 1$: The only difference between the two experiments is whether $(y_0^{(i)}[1 - b^{(i)}])_{i\in[Q'']}$ are sampled uniformly at random from $\mathcal{M}$, or are computed as $y_0^{(i)}[1 - b^{(i)}] \leftarrow \mathsf{CPRF.Eval}(\mathsf{pp}_{1-b^{(i)}}, \mathsf{msk}_{1-b^{(i)}}, x^{(i)})$. By design, for all $i \in [Q'']$, it holds that $F(k, x^{(i)}) = b^{(i)}$ (*i.e.* $\overline{F}(k, x^{(i)}) = 1 - b^{(i)}$). Therefore, whether $b^{(i)} = 0$ or $b^{(i)} = 1$, $x^{(i)}$ is unauthorised for the key $\mathsf{ck}_{1-b^{(i)}}$. By invoking CPRF security twice (once with respect to msk_0 and one with respect to msk_1), $|\Pr[\mathsf{Exp}^{\mathsf{w\text{-}sec}}_{\mathcal{A},Q'',1,0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{w\text{-}sec}}_{\mathcal{A},Q'',1,1}(\lambda) = 1]| \leq 2\epsilon'$.*

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

□

PCF for OT from 1-in-2 Pseudorandomly Constrained PRF.

1-in-2 Pseudorandomly Constrained PRF.

Definition 12 (1-in-2 Pseudorandomly Constrained PRF). *A* weakly/strongly pseudorandomly constrained PRF (PR-CPRF) *is a constrained PRF with domain $\{0,1\}^{n(\lambda)}$ that supports the class of constraints of the form:*

$$\mathcal{C}_\lambda = \left\{ \begin{array}{ll} C_k\colon & \{0,1\}^{n(\lambda)} \to \{0,1\} \\ & x_0 \dots x_{n-1} \mapsto F_{\mathsf{in}}(k, x_0 \dots x_{n-2}) \oplus x_{n-1} \end{array} : k \in \mathcal{K}_\lambda \right\}$$

where $F_{\mathsf{in}}\colon \mathcal{K}_\lambda \times \{0,1\}^{n(\lambda)-1} \to \{0,1\}$ is a weak/strong pseudorandom function.

When convenient, we will refer to the constrained PRF as the outer *PRF and F_{in} as the* inner *PRF.*

PCF for OT from 1-out-of-2 Pseudorandomly Constrained PRF.

Theorem 5 (PCF for OT from PR-CPRF). *Assuming the existence of a weakly/strongly 1-in-2 pseudorandomly constrained PRF, the construction of fig. 11 is a pre-computable weak/strong PCF for OT correlations.*

PCF Precomputable PCF for OT from PR-CPRFs

`Requires:` PR-CPRF $= (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ is a weakly/strongly pseudorandomly constrained PRF with input space $\{0,1\}^{n(\lambda)+1}$ and characterized by inner weak/strong PRF $F_{\mathsf{in}}\colon \mathcal{K}_\lambda \times \{0,1\}^{n(\lambda)} \to \{0,1\}$.

$\mathsf{PCF.Gen}_0(1^\lambda)$:

1. Run $(\mathsf{pp}, \mathsf{msk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$.
2. Set and output $k_0 = (\mathsf{pp}, \mathsf{msk})$.

$\mathsf{PCF.Gen}_1(1^\lambda, k_0)$:

1. Sample $k \xleftarrow{\$} \mathcal{K}_\lambda$.
2. Parse $k_0 = (\mathsf{pp}, \mathsf{msk})$.
3. Run $\mathsf{ck} \leftarrow \mathsf{Constrain}(\mathsf{msk}, C_k)$.
4. Set and output $k_1 = (\mathsf{ck}, k, \mathsf{pp})$.

$\mathsf{PCF.Eval}(1^\lambda, \sigma, k_\sigma, x = (x_0 \dots x_{n-1}))$:

1. If $\sigma = 0$:
   (a) Parse $k_0 = (\mathsf{pp}, \mathsf{msk})$.
   (b) Compute $r_0 \leftarrow \mathsf{Eval}(\mathsf{pp}, \mathsf{msk}, x_0 \dots x_{n-1}0)$.
   (c) Compute $r_1 \leftarrow \mathsf{Eval}(\mathsf{pp}, \mathsf{msk}, x_0 \dots x_{n-1}1)$.
   (d) Set and output $y_0 \leftarrow (r_0, r_1)$.
2. If $\sigma = 1$:
   (a) Parse $k_1 = (\mathsf{ck}, k, \mathsf{pp})$.
   (b) Compute $b \leftarrow F_{\mathsf{in}}(k, x_0 \dots x_{n-1})$.
   (c) Compute $r \leftarrow \mathsf{CEval}(\mathsf{pp}, \mathsf{ck}, x_0 \dots x_{n-1}b)$.
   (d) Run and output $y_1 \leftarrow (b, r)$.

Fig. 11. Pre-Computable wPCF/PCF for OT from weakly/strongly PR-CPRF.

5.2 Pseudorandom Constraints Expressed as IPM

As discussed in section section 4, the Naor-Reingold PRF can be adapted into a CPRF supporting inner-product membership constraints. In this section, we show that the class of "inner-product membership" is expressive enough to capture the various notions of pseudorandom constraints used in section 5.1. As a direct corollary, this yields PCFs for OT correlations from the Naor-Reingold CPRF.

IPM-wPRF. We define the notion of "inner-product membership (weak) PRF" (IPM-PRF) as a (weak) PRF whose evaluation is performed by checking whether the inner product between (some public function of) the key and (some public function of) the input belongs to some public set.

Definition 13 (Inner-Product Membership weak PRF, IPMwPRF). Let $\ell(\cdot)\colon \mathbb{N}^\star \to \mathbb{N}^\star$. Let $<\cdot,\cdot>\colon \mathbb{Z}^{\ell(\lambda)} \times \mathbb{Z}^{\ell(\lambda)} \to \mathbb{Z}$ denote the usual inner-product over $\mathbb{Z}^{n(\lambda)}$. An *Inner-Product Membership weak Pseudorandom Function* ($\mathsf{IPM\text{-}wPRF}$) $F_\lambda\colon \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \{0,1\}$ is one parameterised by a collection $\{S_\lambda\}_{\lambda\in\mathbb{N}}$ of polynomial-size subsets of $\mathbb{Z}$, and two collections of functions $\{f_\lambda\colon \mathcal{X}_\lambda \to \mathbb{Z}_M^{\ell(\lambda)}\}_{\lambda\in\mathbb{N}}$ and $\{g_\lambda\colon \mathcal{K}_\lambda \to \mathbb{Z}_M^{\ell(\lambda)}\}_{\lambda\in\mathbb{N}}$ (where $M(\lambda)$ is some polynomial-size bound we assume to be included in the descriptions of the f_λ and the g_λ) which define it semantically in the following way:

$$\forall\lambda \in \mathbb{N}, \forall k \in \mathcal{K}_\lambda, \forall x \in \mathcal{X}_\lambda, F_\lambda(k,x) = \begin{cases} 0 & \text{if } <f_\lambda(x), g_\lambda(k)> \in S_\lambda \\ 1 & \text{otherwise.} \end{cases}$$

We say that an $\mathsf{IPM\text{-}wPRF}$ is (Q, T, ϵ)-secure if it is a (Q, T, ϵ)-secure weak PRF.

Remark 3 (The opposite of an IPM-wPRF is an IPM-wPRF). The opposite of an IPM-wPRF F_λ parameterised by $\{S_\lambda\}_{\lambda\in\mathbb{N}}$, $\{f_\lambda\}_{\lambda\in\mathbb{N}}$, and $\{g_\lambda\}_{\lambda\in\mathbb{N}}$ is the IPM-wPRF parameterised by $\{[0, \ell(\lambda) \cdot M^2(\lambda)] \setminus S_\lambda\}_{\lambda\in\mathbb{N}}$, $\{f_\lambda\}_{\lambda\in\mathbb{N}}$, and $\{g_\lambda\}_{\lambda\in\mathbb{N}}$. In particular, the existence of a CPRF supporting as constraint $\{F(k,\cdot)\colon k \in \mathcal{K}_\lambda\}$ is equivalent to that of a CPRF supporting as constraint $\{F(k,\cdot)\colon k \in \mathcal{K}_\lambda\} \cup \{1 - F(k,\cdot)\colon k \in \mathcal{K}_\lambda\}$.

One of the downsides of the PCF construction of fig. 9 is that it requires the ability to constrain a CPRF both with respect to a (w)PRF and its opposite. Remark 3 implies that a CPRF supporting the class of inner-product membership constraints can be used in conjunction with an IPM-wPRF to instantiate the template of fig. 9.

Antiperiodic Optimization. In this section we introduce a sufficient condition for a IPM-wPRFs to yield a "1-in-2 pseudorandom selection circuit" which can itself be expressed as inner-product membership. Such IPM-wPRFs are well-suited to instantiate the template of fig. 10.

Definition 14 (Antiperiodic[13] Set). *Let $\Delta \in \mathbb{N}^\star$ and $q \in \mathbb{N}^\star$. We say a bounded set of integers $S \subseteq \mathbb{Z}_q$ is* Δ-antiperiodic *if for all $n \in \mathbb{Z}_q$, $n \in S \Leftrightarrow (n+\Delta) \notin S$. We say a set $S \subseteq \mathbb{Z}_q$ is* antiperiodic *if there exists $\delta \in \mathbb{N}^\star$ such that S is δ-antiperiodic.*

[13] The rationale for the terminology is that a, up to defining booleans as $\{-1, +1\}$, a set is (Δ-)antiperiodic if and only its indicator function is. Recall a function f is Δ-antiperiodic if $\forall x \in D_f, f(x + \Delta) \neq f(x)$.

Definition 15 (Antiperiodic IPM-wPRF). *Let $\Delta \in \mathbb{N}^\star$. We say an IPM-wPRF F, parameterised by the collection of sets $\{S_\lambda\}_{\lambda\in\mathbb{N}^\star}$, is ($\Delta$-)antiperiodic if for all $\lambda \in \mathbb{N}^\star$, S_λ is.*

Observation 1 (Antiperiodic Optimization) *Let $\Delta \in \mathbb{N}^\star$. If F is a Δ-antiperiodic IPM-wPRF (parameterised by $f_\lambda, g_\lambda, S_\lambda$) then for all $k \in \mathcal{K}_\lambda$, the circuit $C_k \colon \{0,1\}^{n(\lambda)} \to \{0,1\}$, $x_0 \dots x_{n-1} \mapsto F(k, x_0 \dots x_{n-2}) \oplus x_{n-1}$ satisfies:*

$$C_k(x) = 0 \Leftrightarrow \langle (f_\lambda(x_0, \dots, x_{n-2}) \| x_{n-1}), (g_\lambda(k) \| \delta) \rangle \in S_\lambda \ .$$

In particular, assuming the existence of an antiperiodic IPM-wPRF and that of a CPRF supporting inner-product membership constraints, there exists a 1-in-2 pseudorandomly constrained PRF (definition 12).

Proof.

$$\begin{aligned} C_k(x) = 0 &\Leftrightarrow F(k, x_0 \dots x_{n-2}) = x_{n-1} \\ &\Leftrightarrow F(k, x_0 \dots x_{n-2}) = x_{n-1} = 0 \vee F(k, x_0 \dots x_{n-2}) = x_{n-1} = 1 \\ &\Leftrightarrow [\langle f_\lambda(x_0, \dots, x_{n-2}), g_\lambda(k) \rangle \in S_\lambda \ \wedge \ x_{n-1} = 0] \\ &\quad \vee [\langle f_\lambda(x_0, \dots, x_{n-2}), g_\lambda(k) \rangle \notin S_\lambda \ \wedge \ x_{n-1} = 1] \\ &\Leftrightarrow \langle (f_\lambda(x_0, \dots, x_{n-2}) \| x_{n-1}), (g_\lambda(k) \| \delta) \rangle \in S_\lambda \ . \end{aligned}$$

The second part of the observation, regarding the existence of a PCF, is a direct corollary of theorem 5. □

5.3 Candidate IPM-wPRF

LWR-based IPM-wPRF candidates. Banerjee, Peikert, and Rosen [BPR12] provided the following weak PRF candidate, whose security can be reduced to the LWR assumption:

$F_{\mathbf{k}}(x) \coloneqq \lfloor <\mathbf{k}, x> \rceil_p$, where $\mathbf{k}, x \in \mathbb{Z}_q^n$ and $q >> p$ ($\lfloor X \rceil_p$ denotes the closest integer to $X \cdot p/q$).

Lemma 1 (IPM-wPRF from Poly-Modulus LWR). *Let q be a polynomial-size modulus. Assuming $\mathsf{LWR}_{q,2}$, there is an IPM-wPRF whose membership set S_λ has size $\lceil q/2 \rceil$.*

Proof. If q is polynomial and $p = 2$, then the candidate wPRF can be cast as an IPM-wPRF parameterised by $\ell(\lambda) \coloneqq n(\lambda)$, $S_\lambda \coloneqq \mathbb{Z}_q \cap p\mathbb{Z}$ (which is indeed polynomial-size when q is polynomial). □

Boneh, Ishai, Passelègue, Sahai, and Wu [BIP+18] introduced a candidate weak PRF which can be seen as a special instance of LWR with constant-size composite modulus. The BIPSW wPRF is defined as:

$$F_{\mathbf{k}}(x) \coloneqq \sum_{i\in[n]} k_i x_i \bmod 2 + \sum_{i\in[n]} k_i x_i \bmod 3 \pmod 2 \ .$$

Lemma 2 (The Boneh-Ishai-Passelègue-Sahai-Wu IPM-wPRF is Antiperiodic). *The BIPSW wPRF can be cast as a 3-antiperiodic IPM-wPRF whose membership set S_λ has size $\lceil \frac{n}{2} \rceil$.*

Proof. Observe that $F_{\mathbf{k}}(x) = 0$ if and only if $<k, x> \bmod 6 \in \{3, 4, 5\}$, and therefore this can be cast as an IPM-wPRF where $\ell(\lambda) \coloneqq n(\lambda)$, $S_\lambda \coloneqq [0, n(\lambda)] \cap (6\mathbb{Z} + \{0, 1, 2\})$. □

IPM-wPRF based on random CSPs. Let $d = \Theta(\log n)$, and let $P \colon \{0,1\}^d \to \{0,1\}$ be an appropriately chosen balanced predicate. Let $\mathsf{in} \coloneqq \lceil \log(n!/d!) \rceil$ (this will be the input length of the wPRF), and consider any polytime-computable function $I. \colon \{0,1\}^{\mathsf{in}} \to A^d_{[n]}, x \mapsto I_x$, where $A^d_{[n]} \subseteq [n]^d$ denotes the set of all arrangements (*i.e.* ordered subsets taken without repetitions) of d elements from $[n]$. We require that every element of $A^d_{[n]}$ admits a unique pre-image by $I.$, but $I.$ is allowed to be undefined for some elements of $\{0,1\}^{\mathsf{in}}$. Such a function exists, and will be viewed as input-preprocessing allowing us to interpret a bit string as arrangement of d elements of $[n]$. The Goldreich-Applebaum-Raykov wPRF [Gol00, AR16] can be defined by:

$$\begin{aligned} F \colon \{0,1\}^n \times \{0,1\}^{\mathsf{in}} &\to \{0,1\} \\ (k, x) &\mapsto P(k_{I_x[0]}, \dots, k_{I_x[d-1]}) \end{aligned} \ .$$

A predicate of particular interest is the XOR-MAJ predicate [Gol00, AL16]. Define $\mathsf{MAJ}_s\colon \{0,1\}^s \to \{0,1\}, (x_1,\ldots,x_s) \mapsto (\sum_{i=1}^{s} x_i \geq s/2)$ and $\mathsf{XOR\text{-}MAJ}_{s,t}\colon \{0,1\}^{s+t} \to \{0,1\}, (x_1,\ldots,x_{s+t}) \mapsto x_1 \oplus \cdots \oplus x_s \oplus \mathsf{MAJ}_\ell(x_{s+1},\ldots,x_{s+t})$. The "Goldreich-Applebaum-Raykov wPRF with d-local $\mathsf{XOR\text{-}MAJ}_{s,t}$ predicate" is defined as follows:

$$\begin{aligned} F\colon \{0,1\}^n \times \{0,1\}^{\mathsf{in}} &\to \{0,1\} \\ (k,x) &\mapsto \mathsf{XOR\text{-}MAJ}_{s,t}(k_{I_x[0]},\ldots,k_{I_x[s+t-1]}) \end{aligned} .$$

Lemma 3 (The Goldreich-Applebaum-Raykov wPRF is an IPM-wPRF). *The following statements hold:*

1. Goldreich-Applebaum-Raykov wPRF (with any low-locality predicate). *If P is any d-local balanced predicate, the GAR wPRF can be cast as an IPM-wPRF parameterised by a membership set S_λ of size 2^{d-1}.*
2. Goldreich-Applebaum-Raykov wPRF (optimized for the XOR-MAJ predicate). *If P is the $\mathsf{XOR\text{-}MAJ}_{s,t}$ predicate, the GAR wPRF can be cast as an IPM-wPRF parameterised by a membership set S_λ of size $2 \cdot \lceil s/2 \rceil \cdot \lceil t/2 \rceil$.*

Proof.

1. *Let $\mathsf{Supp}(P) \coloneqq \{v \in \mathbb{Z}_{2^d} : P(\mathsf{IntToBitString}(v)) = 1\}$. Observe that:*

$$\begin{aligned} F(k,x) = 0 &\Leftrightarrow P(k_{I_x[1]},\ldots,k_{I_x[d]}) = 0 \\ &\Leftrightarrow \mathsf{BitStringToInt}(k_{I_x[0]},\ldots,k_{I_x[d-1]}) \notin \mathsf{Supp}(P) \\ &\Leftrightarrow k_{I_x[0]} \cdot 1 + \cdots + k_{I_x[d-1]} \cdot 2^{d-1} \notin \mathsf{Supp}(P) \\ &\Leftrightarrow <k, f(x)> \in \mathbb{Z}_{2^d} \setminus \mathsf{Supp}(P) \end{aligned} \tag{2}$$

where $f\colon \{0,1\}^{\mathsf{in}} \to \mathbb{Z}_{2^d}^n,\ x \mapsto \sum_{i=0}^{d-1} 2^i \cdot \mathbf{1}_{I_x[i]}$.

The Goldreich-Applebaum-Raykov wPRF can therefore be cast as an IPM-wPRF parameterised by $\ell \coloneqq n$, $S_\lambda \coloneqq \mathbb{Z}_{2^d} \setminus \mathsf{Supp}(P)$, $f_\lambda \coloneqq f$, and $g_\lambda \coloneqq \mathsf{Id}$. Note that because P is a balanced predicate, $|\mathsf{Supp}(P)| = 2^{d-1}$, and that because $d = \Theta(\log n)$, $|S_\lambda| = n^{\mathcal{O}(1)}$.

2. *Observe that:*

$$\begin{aligned} F_k(x) = 0 &\Leftrightarrow \mathsf{XOR}(k_{I_x[0]},\ldots,k_{I_x[s-1]}) = \mathsf{MAJ}_t(k_{I_x[s]},\ldots,k_{I_x[s+t-1]}) \\ &\Leftrightarrow \left[\mathsf{XOR}(k_{I_x[0]},\ldots,k_{I_x[s-1]}) = 0 \ \wedge\ \mathsf{MAJ}_t(k_{I_x[s]},\ldots,k_{I_x[s+t-1]}) = 0\right] \\ &\quad \vee \left[\mathsf{XOR}(k_{I_x[0]},\ldots,k_{I_x[s-1]}) = 1 \ \wedge\ \mathsf{MAJ}_t(k_{I_x[s},\ldots,k_{x[s+t-1]}) = 1\right] . \end{aligned} \tag{3}$$

Further observe that:

$$\begin{aligned} \mathsf{XOR}(k_{I_x[0]},\ldots,k_{I_x[s-1]}) &= \left[<1^s, (k_{I_x[0]},\ldots,k_{I_x[s-1]})> \bmod 2\right] \\ &= \left[<\mathbf{1}_{I_x[0,s-1]}, k> \bmod 2\right] \end{aligned} \tag{4}$$

and that

$$\begin{aligned} \mathsf{MAJ}_t(k_{I_x[s]},\ldots,k_{I_x[s+t-1]}) &= \left[(<1^t, (k_{I_x[s]},\ldots,k_{I_x[s+t-1]})>) \geq t/2\right] \\ &= \left[<\mathbf{1}_{I_x[s,s+t-1]}, k> \geq\ t/2\right] . \end{aligned} \tag{5}$$

By combining eqs. (3) to (5), we obtain:

$$\begin{aligned} F_k(x) = 1 &\Leftrightarrow \left[<k, \mathbf{1}_{I_x[0,s-1]}> \equiv 0 \bmod 2 \ \wedge\ \ <k, \mathbf{1}_{I_x[s,s+t-1]}> < t/2\right] \\ &\quad \vee \left[<k, \mathbf{1}_{I_x[0,s-1]}> \equiv 1 \bmod 2 \ \wedge\ \ <k, \mathbf{1}_{I_x[s,s+t-1]}> \geq t/2\right] \\ &\Leftrightarrow \left[<\mathbf{1}_{I_x[0,s-1]} + (t+1) \cdot \mathbf{1}_{I_x[s,s+t-1]}, k> \in\ \ S_{\mathsf{XOR\text{-}MAJ}}\right] \end{aligned}$$

where $S_{\mathsf{XOR\text{-}MAJ}} \coloneqq \{r + (t+1) \cdot q\ :\ (q,r) \in (Q_0 \times R_0) \sqcup (Q_1 \times R_1)\}$
with $Q_0 \coloneqq [0, \lceil s/2 \rceil - 1]$, $Q_1 \coloneqq [\lceil s/2 \rceil, s-1]$,
and $R_b \coloneqq \{r \in [0, t-1] : r \equiv b \bmod 2\}$ (for $b \in \{0,1\}$).

The Goldreich-Applebaum-Raykov wPRF with XOR-MAJ predicate can therefore be cast as an IPM-wPRF parameterised by $\ell := n$, $S_\lambda := S_{\mathsf{XOR\text{-}MAJ}}$, $f_\lambda \colon \{0,1\}^{\mathsf{in}} \to \{0,1\}^n, x \mapsto \mathbf{1}_{x[0,s-1]} + (t+1) \cdot \mathbf{1}_{x[s,s+t-1]}$ *(in other words,* f_λ *maps* x *to the length-*n *vector with* 1 *in positions* $I_x[0], \ldots, I_x[s-1]$*, with* $(t+1)$ *in positions* $I_x[s], \ldots, I_x[s+t-1]$*, and with* 0 *everywhere else), and* $g_\lambda := \mathsf{Id}$ *. Finally, note that* $|S_{\mathsf{XOR\text{-}MAJ}}| = |Q_0| \cdot |R_0| + |Q_1| \cdot |R_1| \leq 2 \cdot \lceil \frac{s}{2} \rceil \cdot \lceil \frac{t}{2} \rceil$.

□

As described in the proof of lemma 3, the Goldreich-Applebaum-Raykov IPM-wPRF with $\mathsf{XOR\text{-}MAJ}_{s,t}$ predicate is almost $(t+1)$-antiperiodic. The set $S_{\mathsf{XOR\text{-}MAJ}}$ is represented in fig. 12.

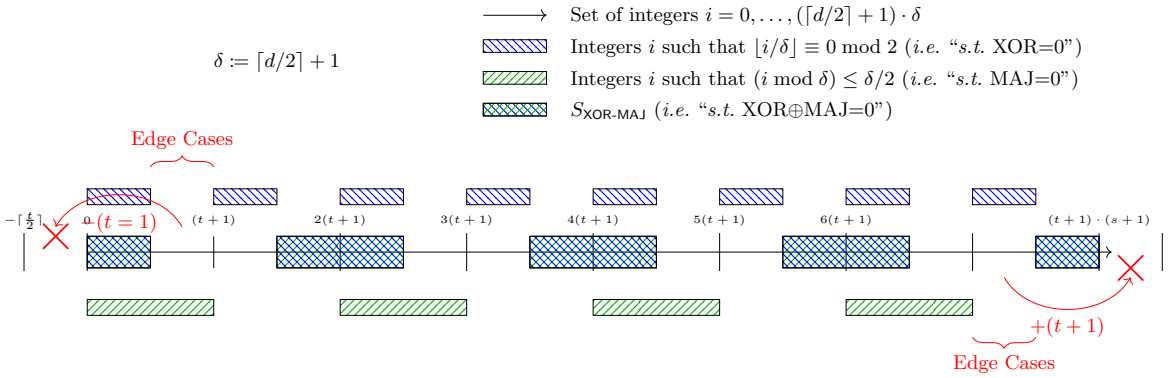

Fig. 12. Representation of the (inner-product) membership set of the GAR IPM-wPRF with XOR-MAJ predicate. Considering the membership set $[-\lceil \frac{t}{2} \rceil, 0] \cup S_{\mathsf{XOR\text{-}MAJ}} \cup [(t+1)\cdot(s+1), (t+1)\cdot(s+1) + \lfloor \frac{t}{2} \rfloor]$ yields a δ-antiperiodic IPM-wPRF.

Lemma 4 (The XOR-MAJ Goldreich-Applebaum-Raykov IPM-wPRF is Antiperiodic). *The Goldreich-Applebaum-Raykov IPM-wPRF with predicate:* $\mathsf{XOR\text{-}MAJ}_{s,t}$ *can be cast as a* $(t+1)$*-antiperiodic IPM-wPRF parameterised by a membership set of size* $2 \cdot \lceil s/2 \rceil \cdot \lceil t/2 \rceil + (t+1)$.

5.4 Distributed Interactive Key Generation via MPC

In this section, we show that our PCF for OT correlations based on the Naor-Reingold PRF admits an efficient protocol for securely distributing the key-generation phase.

Definitions. We start by introducing in figs. 13 and 14 the ideal functionalities for the distributed key-generation and for 1-in-2 oblivious transfer.

Functionality PCF Distributed Key-Generation $\mathcal{F}_{\mathsf{DKG}}$

Parameters: The ideal functionality Fsd is parameterised by a number of parties $N \geq 2$, and an N-party PCF $\mathsf{PCF} = (\mathsf{PCF.Gen}, \mathsf{PCF.Eval})$ (for some correlation).

$\mathcal{F}_{\mathsf{DKG}}$ interacts with parties $P_1, \ldots, P_N$ in the following manner.

Output: Run $(k_1, \ldots, k_N) \xleftarrow{\$} \mathsf{PCF.Gen}(1^\lambda)$; Output k_i to each party P_i (for $1 \leq i \leq N$).

Fig. 13. Ideal functionality $\mathcal{F}_{\mathsf{DKG}}$ for the distributed generation of PCF keys.

Functionality Oblivious Transfer $\mathcal{F}_{\mathsf{OT}}$

The functionality $\mathcal{F}_{\mathsf{OT}}$ for 1-out-of-2 oblivious transfer is parameterised by a message space $\mathcal{M}$, and interacts with two parties S (the sender) and R (the receiver).

Input: Wait to receive $(\texttt{sender}, \mathsf{sid}, m_0, m_1)$ (where $m_0, m_1 \in \mathcal{M}$) from S and $(\texttt{receiver}, \mathsf{sid}, b)$ (where $b \in \{0,1\}$) from R.
Output: Send (sid, m_b) to R and (sid) to S, then halt.

Fig. 14. Ideal functionality $\mathcal{F}_{\mathsf{OT}}$ for 1-out-of-2 oblivious transfer.

The Protocol. The protocol, provided in fig. 15, revolves around the following observation. The key difficulty in generating the keys is that P_1, holding $\mathbf{z} \in \{0,1\}^n$, must retrieve $(r^{-z_i} \cdot \alpha_i)_{i\in[n]}$ from P_0, without learning $(r^{-(1-z_i)} \cdot \alpha_i)_{i\in[n]}$. Our observation is that this can be performed efficiently using 1-out-of-2 OT, where P_0 acts as sender with messages $(\alpha_i, r^{-z_i}\alpha_i)$ and P_1 acts as receiver with selection bit z_i.

Protocol PCF Distributed Key-Generation Π_{DKG} for the PCF of fig. 11 (parameterised by the CPRF of fig. 7)

Parties: P_0, P_1.

Parameters:

- $\mathsf{sid}_1, \ldots, \mathsf{sid}_n$ are n distinct session ids.
- *Parameters for the CPRF (fig. 11):* $G\colon \{0,1\}^\lambda \to \{0,1\}^{(n+2)\lambda}$ and $G'\colon \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ are PRGs.
- *Parameters for the PCF (fig. 7):* $F \coloneqq \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \mathcal{M}_\lambda$ is a δ-antiperiodic $\mathsf{IPM\text{-}wPRF}$ parameterised by membership set S_λ, input-preprocessing function $f_\lambda\colon \mathcal{X}_\lambda \to \{0,1\}^n$, and key-preprocessing function $g_\lambda\colon \mathcal{K}_\lambda \to \{0,1\}^n$.

Hybrid Model: The protocol is defined in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model.

The Protocol:

1. ***Preprocessing Phase:***
   - P_0 does the following:
     (a) Run $(\mathbb{G}, g, p) \xleftarrow{\$} \mathsf{GenPar}(1^\lambda)$.
     (b) Sample $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$.
     (c) Set $\mathsf{msk} \leftarrow \mathsf{seed}$, $\mathsf{pp} \leftarrow (\mathbb{G}, g, p)$, and $k_0 \leftarrow (\mathsf{pp}, \mathsf{msk})$.
     (d) Compute $(a_0, \mathsf{seed}_1, \ldots, \mathsf{seed}_{n+1}) \leftarrow G(\mathsf{seed})$
     (e) Sample $r \xleftarrow{\$} \mathbb{F}_p^\star$
     (f) For $s \in S$, compute $g_s \leftarrow g^{a_0 \cdot r^s}$
     (g) For $i \in [n]$:
       - Set $\tilde{\alpha}_i^{(0)} \coloneqq \mathsf{seed}_i$
       - Set $\tilde{\alpha}_i^{(1)} \coloneqq r^{-1} \cdot G'(\mathsf{seed}_i)$ [a]
     (h) Set $\tilde{\alpha}_{n+1}^{(1)} \coloneqq r^{-\delta} \cdot G'(\mathsf{seed}_{n+1})$
   - P_1 does the following:
     (a) Sample $k \xleftarrow{\$} \mathcal{K}_\lambda$
     (b) Set $\mathbf{z} = (z_1, \ldots, z_{n+1}) \leftarrow (g_\lambda(k), \delta)$
2. ***Interactive Phase:***
   - P_0 sends $(g_s)_{s\in S}$ to P_1

- For $i = 1 \dots n$:
  - P_0 sends $(\mathsf{sender}, \mathsf{sid}_i, \tilde{\alpha}_i^{(0)}, \tilde{\alpha}_i^{(1)})$ to $\mathcal{F}_{\mathsf{OT}}$.
  - P_1 sends $(\mathsf{receiver}, \mathsf{sid}_i, z_i)$ to $\mathcal{F}_{\mathsf{OT}}$ and waits to receive (sid_i, m_i).
- P_0 sends $\tilde{\alpha}_{n+1}^{(1)}$ to P_1
- P_0 sends pp to P_1

3. ***Output Phase:***
   - P_1 sets $\mathsf{ck} \leftarrow (m_1, \dots, m_n, \tilde{\alpha}_{n+1}^{(1)}, (g_s)_{s \in S}, \mathbf{z})$ and $k_1 \leftarrow (\mathsf{ck}, k, \mathsf{pp})$
   - P_0 outputs k_0
   - P_1 output k_1

[a] $G'(\mathsf{seed}_i)$ is viewed as an element of $\mathbb{F}_p$; p is chosen so that $\log p = 2\lambda$.

Fig. 15. Two-party protocol for securely generating the keys of the PCF from fig. 11 (instantiated with the Naor-Reingold CPRF fig. 7).

Theorem 6 (Secure PCF Key Generation). *The protocol of fig. 15 securely perfectly UC-securely realises the two-party functionality fig. 13 (as parameterised by the PCF of fig. 11, in turn instantiated with the CPRF of fig. 7) in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model, in the presence of a passive adversary statistically corrupting at most one of the parties. The protocol makes n calls to the $\mathcal{F}_{\mathsf{OT}}$ functionality and additionally uses $|S| \cdot \log|\mathbb{G}| + 2\lambda + \log|\mathsf{pp}|$ bits of communication. The protocol requires a single round of communication (in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model).*

Proof.

- Security against P_0. *Perfect security against a corrupted P_0 follow from the fact they receive no messages (either from P_1 or from $\mathcal{F}_{\mathsf{OT}}$) as well as the observation that the code defining P_0's output k_0 can be parsed as being code-equivalent to "$k_0 \xleftarrow{\$} \mathsf{PCF.Gen}_0(1^\lambda)$" (which is how the ideal functionality fig. 13 computes it). P_0's view from the protocol can therefore be perfectly simulated.*
- Security against P_1. *To show perfect security against a corrupted P_1, let $\mathcal{A}$ be a semi-honest, static, adversary that interacts with parties P_0, P_1 running the protocol in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model. Consider the following simulator $\mathcal{S}$, which runs internally a copy of $\mathcal{A}$.*
  - **Simulating the communication with the environment $\mathcal{Z}$:** *$\mathcal{S}$ writes every input value it receives from $\mathcal{Z}$ on $\mathcal{A}$'s input tape (as if coming from $\mathcal{A}$'s environment), and copies every value on $\mathcal{A}$'s output tape to $\mathcal{S}$'s own output tape (to be read by $\mathcal{Z}$).*
  - **Simulating the protocol's execution:**
    1. *$\mathcal{S}$ activates $\mathcal{A}$.*
    2. *$\mathcal{S}$ waits to receive k_1 from $\mathcal{F}_{\mathsf{DKG}}$ (on behalf of P_1).*
    3. *$\mathcal{S}$ parses k_1 as $k_1 = (\mathsf{ck}, k, \mathsf{pp})$.*
    4. *$\mathcal{S}$ parses ck as $\mathsf{ck} = (m_1, \dots, m_n, \tilde{\alpha}_{n+1}^{(1)}, (g_s)_{s \in S}, \mathbf{z})$.*
    5. *$\mathcal{S}$ writes $(g_s)_{s \in S}$ on $\mathcal{A}$'s input tape (as if P_0 sent this to P_1).*
    6. *$\mathcal{S}$ waits to read on $\mathcal{A}$'s output tape the messagesthat $\mathcal{A}$ would send to $\mathcal{F}_{\mathsf{OT}}$ (on behalf of P_1), then writes $(\mathsf{sid}_i, m_i)_{i \in [n]}$ on $\mathcal{A}$'s input tape (as if $\mathcal{F}_{\mathsf{OT}}$ sent this to P_1).*
    7. *$\mathcal{S}$ writes $\tilde{\alpha}_{n+1}^{(1)}$ on $\mathcal{A}$'s input tape (as if P_0 sent this to P_1).*
    8. *$\mathcal{S}$ writes pp on $\mathcal{A}$'s input tape (as if P_0 sent this to P_1).*
- Protocol's efficiency. *The total communication of the protocol is the following: P_0 sends $(g_s)_{s \in S}$ (each g_s is a group element of $\mathbb{G}$), $\tilde{\alpha}_{n+1}^{(1)}$ (which is an element of $\mathbb{F}_p$, hence 2λ-bit long), and pp.*

□

6 Public-Key PCF for OT Correlations

6.1 Formal Definition

Here, we introduce and formalize the notion of public-key pseudorandom correlation function (PK-PCF). The main property of a public-key PCF is the non-interactive generation of evaluation

keys. More precisely, in a PK-PCF, the generation of evaluation keys is done in two separate steps: public key generation $\mathsf{PCF.Gen}$, and evaluation key derivation $\mathsf{PCF.KeyDer}$. Let $\sigma \in \{0,1\}$ denote the index of a party in a PK-PCF protocol. In the first step, each party locally runs $\mathsf{PCF.Gen}(\sigma)$ to generate a key pair $(\mathsf{sk}_\sigma, \mathsf{pk}_\sigma)$, and then broadcasts the public key pk_σ. In the second step, each party uses the secret key sk_σ and the public key of the other party $\mathsf{pk}_{1-\sigma}$ and runs the $\mathsf{PCF.KeyDer}$ algorithm in order to derive the PCF evaluation key k_σ. After this step, similarly to an interactive PCF, parties can use their keys k_σ and run the evaluation algorithm $\mathsf{PCF.Eval}(\sigma, k_\sigma, x)$ to compute a correlated output y_σ on an input x.

In the following, we state the formal definition and security requirements of a *weak* PK-PCF.

Definition 16 (Public-Key Pseudorandom Correlation Function (PK-PCF)). *Let $\mathcal{Y}$ be a reverse-sampleable correlation with output length functions $\ell_0(\lambda), \ell_1(\lambda)$ and let $\lambda \le n(\lambda) \le \mathsf{poly}(\lambda)$ be an input length function. A PK-PCF consists of the following four polynomial algorithms:*

- $\mathsf{PCF.Setup}(1^\lambda)$: A probabilistic algorithm that on input 1^λ, outputs a public parameter pp. For simplicity of notation, we assume that all other algorithms have access to pp.
- $\mathsf{PCF.Gen}(\sigma)$: A probabilistic algorithm that on input $\sigma \in \{0,1\}$, outputs a pair of public key and secret key $(\mathsf{sk}_\sigma, \mathsf{pk}_\sigma)$.
- $\mathsf{PCF.KeyDer}(\sigma, \mathsf{sk}_\sigma, \mathsf{pk}_{1-\sigma})$: A deterministic algorithm that on input $\sigma \in \{0,1\}$, a secret key sk_σ and a public key $\mathsf{pk}_{1-\sigma}$, outputs an evaluation key k_σ.
- $\mathsf{PCF.Eval}(\sigma, k_\sigma, x)$: A deterministic algorithm that on input $\sigma \in \{0,1\}$, a key k_σ and input value $x \in \{0,1\}^{n(\lambda)}$, outputs $y_\sigma \in \{0,1\}^{\ell_\sigma(\lambda)}$.

We say that a PK-PCF for a correlation $\mathcal{Y}$ is (N, B, ϵ)-secure, if the following two conditions hold:

Pseudorandom $\mathcal{Y}$-correlated outputs. For every non-uniform adversary $\mathcal{A}$ of size $B(\lambda)$, it holds that for all sufficiently large λ,

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

where $\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},N,b}$ ($b \in \{0,1\}$) is defined as in Figure 16. In particular, the adversary is given access to $N(\lambda)$ samples.

Experiment Pseudorandom $\mathcal{Y}$-Correlated Outputs

$\underline{\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},N,0}(\lambda)}$:

$\mathsf{pp} \leftarrow \mathsf{PCF.Setup}(1^\lambda)$
For $\sigma = 0, 1$**:**
  $(\mathsf{sk}_\sigma, \mathsf{pk}_\sigma) \leftarrow \mathsf{PCF.Gen}(\sigma)$
For $i = 1, \ldots, N(\lambda)$:
  $x^{(i)} \xleftarrow{\$} \{0,1\}^{n(\lambda)}$
  $(y_0^{(i)}, y_1^{(i)}) \leftarrow \mathcal{Y}(1^\lambda)$
$b \leftarrow \mathcal{A}(1^\lambda, \mathsf{pk}_0, \mathsf{pk}_1, (x^{(i)}, y_0^{(i)}, y_1^{(i)})_{i \in [N(\lambda)]})$
Output b

$\underline{\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},N,1}(\lambda)}$:

$\mathsf{pp} \leftarrow \mathsf{PCF.Setup}(1^\lambda)$
For $\sigma = 0, 1$**:**
  $(\mathsf{sk}_\sigma, \mathsf{pk}_\sigma) \leftarrow \mathsf{PCF.Gen}(1^\lambda, \mathsf{pp}, \sigma)$
For $\sigma = 0, 1$**:**
  $k_\sigma \leftarrow \mathsf{PCF.KeyDer}(\sigma, \mathsf{sk}_\sigma, \mathsf{pk}_{1-\sigma})$
For $i = 1, \ldots, N(\lambda)$:
  $x^{(i)} \xleftarrow{\$} \{0,1\}^{n(\lambda)}$
  $y_0^{(i)} \leftarrow \mathsf{PCF.Eval}(0, k_0, x^{(i)})$
  $y_1^{(i)} \leftarrow \mathsf{PCF.Eval}(1, k_1, x^{(i)})$
$b \leftarrow \mathcal{A}(1^\lambda, \mathsf{pk}_0, \mathsf{pk}_1, (x^{(i)}, y_0^{(i)}, y_1^{(i)})_{i \in [N(\lambda)]})$
Output b

Fig. 16. Pseudorandom $\mathcal{Y}$-correlated outputs of a PK-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,

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

where $\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},N,\sigma,b}$ ($b \in \{0,1\}$) is defined as in Figure 17. In particular, the adversary is given access to $N(\lambda)$ samples

Experiment PCF Security

$\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},N,\sigma,0}(\lambda)$:

$\mathsf{pp} \leftarrow \mathsf{PCF.Setup}(1^\lambda)$
For $\hat{\sigma} = 0, 1$**:**
  $(\mathsf{sk}_{\hat{\sigma}}, \mathsf{pk}_{\hat{\sigma}}) \leftarrow \mathsf{PCF.Gen}(\hat{\sigma})$
$k_{1-\sigma} \leftarrow \mathsf{PCF.KeyDer}(1-\sigma, \mathsf{sk}_{1-\sigma}, \mathsf{pk}_\sigma)$
For $i = 1, \ldots, N(\lambda)$:
  $x^{(i)} \xleftarrow{\$} \{0,1\}^{n(\lambda)}$
  $y^{(i)}_{1-\sigma} \leftarrow \mathsf{PCF.Eval}(1-\sigma, k_{1-\sigma}, x^{(i)})$
Let $T = (x^{(i)}, y^{(i)}_{1-\sigma})_{i\in[N(\lambda)]}$
$b \leftarrow \mathcal{A}(1^\lambda, \mathsf{pk}_0, \mathsf{pk}_1, \sigma, \mathsf{sk}_\sigma, T)$
Output b

$\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},N,\sigma,1}(\lambda)$:

$\mathsf{pp} \leftarrow \mathsf{PCF.Setup}(1^\lambda)$
For $\hat{\sigma} = 0, 1$**:**
  $(\mathsf{sk}_{\hat{\sigma}}, \mathsf{pk}_{\hat{\sigma}}) \leftarrow \mathsf{PCF.Gen}(\hat{\sigma})$
$k_\sigma \leftarrow \mathsf{PCF.KeyDer}(\sigma, \mathsf{sk}_\sigma, \mathsf{pk}_{1-\sigma})$
For $i = 1, \ldots, N(\lambda)$:
  $x^{(i)} \xleftarrow{\$} \{0,1\}^{n(\lambda)}$
  $y^{(i)}_{\sigma} \leftarrow \mathsf{PCF.Eval}(\sigma, k_\sigma, x^{(i)})$
  $y^{(i)}_{1-\sigma} \leftarrow \mathsf{RSample}(1^\lambda, \sigma, y^{(i)}_\sigma)$
Let $T = (x^{(i)}, y^{(i)}_{1-\sigma})_{i\in[N(\lambda)]}$
$b \leftarrow \mathcal{A}(1^\lambda, \mathsf{pk}_0, \mathsf{pk}_1, \sigma, \mathsf{sk}_\sigma, T)$
Output b

Fig. 17. Security of a PK-PCF. RSample is the algorithm for reverse sampling $\mathcal{Y}$.

6.2 A Public-Key PCF via Bellare-Micali Non-Interactive OT

In Section 5, we provided transformations from CPRFs supporting (w)PRF constraints and their opposite to PCFs for OT correlations. More specifically, Figure 10 presents a simple (unoptimized) instance of this transformation where in order to generate the PCF keys, two calls to the underlying CPRF are made. In this section, we discuss how to generate the resulting PCF keys of this transformation non-interactively when plugging in our Naor-Reingold CPRF.

Let $F : \mathcal{K} \times \mathcal{X} \to \{0,1\}$ be an IPM-wPRF with respect to an IPM set S and predicate functions f, g (See Definition 13), and let us plug in our CPRF construction (Figure 6) in this transformation. The resulting PCF keys are:

$$k_0 = ((\mathsf{pp}_0, \mathsf{msk}_0), (\mathsf{pp}_1, \mathsf{msk}_1)), \quad k_1 = ((\mathsf{pp}_0, \mathsf{ck}_0), (\mathsf{pp}_1, \mathsf{ck}_1), g(\hat{\mathsf{k}})),$$

where $\mathsf{msk}_b \leftarrow \mathsf{CPRF.KeyGen}(1^\lambda)$, and $\mathsf{ck}_b \leftarrow \mathsf{CPRF.Constrain}(\mathsf{msk}_b, C_b)$, where $C_0(x) = F_{\hat{\mathsf{k}}}(x)$, and $C_1(x) = 1 - F_{\hat{\mathsf{k}}}(x)$ for a random key $\hat{\mathsf{k}} \xleftarrow{\$} \mathcal{K}$, and $b \in \{0,1\}$.

We now take a closer look at the correlation between a master secret key and a constrained key in Naor-Reingold CPRF. When working over a group $\mathbb{G} = \langle g \rangle$ of prime order p, a master secret key is of the form $\mathsf{msk} = (a_0, a_1, \ldots, a_n) \xleftarrow{\$} \mathbb{Z}_p^{n+1}$, and a constrained key for an IPM constraint (k, S) is $\mathsf{ck} = (\boldsymbol{\alpha}, (g_s)_{s\in S}, \mathsf{k})$, where $\alpha_i = a_i \cdot r^{-k_i}$ for all $i \in [n]$, and $g_s = g^{a_0 \cdot r^s}$ for all $s \in S$. Here, $\mathsf{k} = g(\hat{\mathsf{k}})$, where $\hat{\mathsf{k}}$ is the key of the wPRF F.

In order to generate the PCF keys non-interactively, we need to derive a valid pair of msk and ck non-interactively. Given that the set S attributed with the IPM-wPRF is public, party 0 can sample a random element $a_0 \xleftarrow{\$} \mathbb{Z}_p$ and compute and publish the tuple $(g_s)_{s\in S}$. Party 1 can then use this tuple as a part of its PCF key.
The more correlated elements of msk and ck are the vectors $(a_1, \ldots, a_n) \in \mathsf{msk}$ and $(\alpha_1, \ldots, \alpha_n) \in \mathsf{ck}$ that satisfy the equation $\alpha_i = a_i \cdot r^{-g(k)_i}$ for all $i \in [n]$. Note that the wPRF key k is only known to party 1 and the secret elements $(a_1, \ldots, a_n)$ and r are only known to party 0. In what follows, we discuss how to generate two pairs (a, r) and (α, b) such that $\alpha = a \cdot r^b$, where $\alpha, a, r \in \mathbb{Z}_p$ and $b \in \{0,1\}$. We can then extend the solution to generate all the elements of the two vectors $\mathbf{a}$ and $\boldsymbol{\alpha}$.

Bellare-Micali Non-interactive OT [BM90] provides a simple but costly solution by using ElGamal encryption and Pedersen commitment to generate two such correlated pairs (a, r) and (α, b), for a bit $b \in \{0,1\}$ and elements $a, r, \alpha \in \mathsf{QR}_p$, where QR_p denotes the subgroup of quadratic residues modulo p. The details of this protocol can be found in Figure 18.

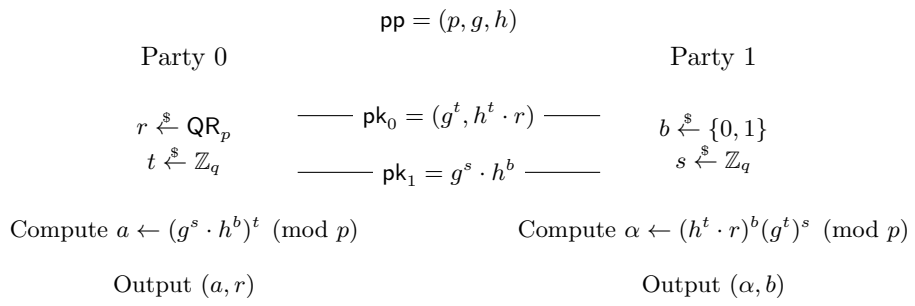

Fig. 18. Bellare-Micali Non-Interactive OT [BM90]. Here, $p = 2q+1$ for primes p and q. The public elements g, h are randomly sampled from QR_p and $\mathsf{DLog}_g(h)$ is unknown to both parties.

Note that in this protocol, the element r should be a quadratic residue modulo p. Therefore, using this protocol for setting up a public key generation for our PCF protocol from the Naor-Reingold CPRF implies assuming the hardness of the sparse power-DDH problem for a quadratic residue $r \in \mathsf{QR}_p$. This variant is implied by the sparse power-DDH assumption. However, for the security of this non-interactive protocol we have to assume the hardness of DDH problem over QR_p which imposes choosing a large-enough prime p due to subexponential-time attacks on DDH over finite fields. This makes the resulting public-key PCF inefficient in terms of both the size of public-key and evaluation keys and computation time.

In the next section, we propose an alternative efficient way of setting up the public keys.

6.3 A Better Construction from Paillier-ElGamal

In this section, we present an efficient public-key PCF construction where in order to derive a pair of OT-correlated evaluation keys, we perform the non-interactive OT protocol of Bellare-Micali [BM90] over a Paillier group. This is in essence the Non-interactive VOLE protocol of [OSY21], followed by additional steps in order to derive the correlated keys over $\mathbb{Z}_p^*$.

Let $N = PQ$ be a Blum integer, meaning that P and Q are prime numbers of the form $P = 2P' + 1$ and $Q = 2Q' + 1$ for λ-bit prime numbers P' and Q'. The key generation and derivation of our public-key PCF work over the group $\mathbb{Z}_{N^2}^* \approx \mathbb{H} \times \mathbb{NR}_N$, where $\mathbb{H} = \{(1+N)^i : i \in [N]\}$ is of order N, and $\mathbb{NR}_N = \{x^N : x \in \mathbb{Z}_{N^2}^*\}$ is of order $\varphi(N)$.

We first recall the following lemma due to [OSY21], where they introduce a distributed discrete logarithm algorithm for a subset of $\mathbb{Z}_{N^2}^*$.

Lemma 5 ([OSY21]). *There exists an algorithm* $\mathsf{DDLog}_N(g)$ *for which the following holds: Let* $g_0, g_1 \in \mathbb{Z}_{N^2}^*$*, such that* $g_0 = g_1(1+N)^x \pmod{N^2}$*. If* $z_0 = \mathsf{DDLog}_N(g_0)$ *and* $z_1 = \mathsf{DDLog}_N(g_1)$*, then* $z_0 - z_1 = x \pmod{N}$*.*

More precisely, $\mathsf{DDLog}_N(g)$ *works as follows:*

- $\mathsf{DDLog}_N(g)$
  - *Write* $g = h + h'N$*, where* $h, h' < N$*, using the division algorithm.*
  - *Output* $z = h'h^{-1} \pmod{N}$*.*

Construction idea: Let g_q be a generator of QR_p, and G, H two random elements of $\mathbb{NR}_{2N}$. Party 0 samples $r' \xleftarrow{\$} \mathbb{Z}_q$ and sets $r := g_q^{r'} \pmod{p}$. It then computes its public key as a Paillier-ElGamal encryption of r', i.e., $\mathsf{pk}_0 = (G^t, H^t \cdot (1+N)^{r'})$. Party 1 holding a bit $k \in \{0,1\}$ computes its public key as a Pedersen commitment of k over $\mathbb{NR}_{2N}$, i.e., $\mathsf{pk}_1 = G^s \cdot H^k$. Following the same computations as in Figure 18, at the end of the protocol, the two parties derive two pairs (A, r) and (B, k), where $B = A \cdot (1+N)^{r' \cdot k}$. In other words, the parties derive multiplicative shares of $(1+N)^{r' \cdot k}$. We now use the DDLog_N algorithm of Lemma 5 to locally convert these multiplicative shares to subtractive shares. More precisely, Party 0 and Party 1 respectively compute $\hat{a} \leftarrow \mathsf{DDLog}_N(A)$ and $\hat{\alpha} \leftarrow \mathsf{DDLog}_N(B)$,

where $\hat{a} - \hat{\alpha} = r' \cdot k \pmod N$. Note that we can furthermore view these elements as the shares of $r' \cdot k$ over the integers, if it holds that $r' \cdot k \ll N$. Therefore, setting $q < N/2^\lambda$, it is now possible for Party 0 and Party 1 to respectively compute $a = g_q^{\hat{a}}$ and $\alpha = g_q^{\hat{\alpha}}$ such that $\alpha = a \cdot r^{-k} \pmod p$. Recall that the goal of the protocol is to derive a pair of Naor-Reingold master secret key and constrained key for a constraint vector $\mathsf{k} \in \{0,1\}^n$ that is a wPRF key. We need to therefore derive n such correlated elements. To do so, it is enough for Party 1 to publish n Pedersen commitments to each bit of its wPRF key k. The details of the construction are presented in Figure 19.

PK-PCF for OT Correlations from sparse power-DDH and DCR

`Requires:`

- Let $F : \mathcal{K} \times \mathcal{X} \to \{0,1\}$ be an IPM-wPRF with respect to an IPM set S and predicate functions f, g (See Definition 13). We assume that S is Δ-antiperiodic, and consider $S' = S \cup \{-\Delta/2, \dots, -1\}$.
- Let p be a safe prime, i.e., $p = 2q + 1$ for a prime q, and $N = PQ$ for primes P and Q.
- Let DDLog_N be the distributed discrete logarithm algorithm described in Lemma 5.
- Let Hash be a hash function modeled as a random oracle.

$\mathsf{PCF.Setup}(1^\lambda)$:

1. $(\mathbb{G}, g, p) \xleftarrow{\$} \mathsf{GenPar}(1^\lambda)$.
2. Sample a generator g_q of QR_p.
3. Sample $G' \xleftarrow{\$} \mathbb{Z}_{N^2}$, and set $G \leftarrow (G')^{2N} \pmod{N^2}$.
4. Sample $d \xleftarrow{\$} \mathbb{Z}_{N^2}$, and set $H \leftarrow G^d \pmod{N^2}$.
5. Output $\mathsf{pp} = (\mathbb{G}, p, g_q, (G, H), F)$.

$\mathsf{PCF.Gen}_0(1^\lambda)$:

1. Sample $h \xleftarrow{\$} \mathbb{G}$.
2. Sample $r' \xleftarrow{\$} \mathbb{Z}_q$, and set $r := g_q^{r'} \pmod p$.
3. Compute $(c_0, c_1) = (G^t, H^t \cdot (1+N)^{r'})$, where $t \xleftarrow{\$} \mathbb{Z}_N$.
   // Paillier-ElGamal encryption of r'.
4. For $s \in S'$, compute and set $h_s := h^{r^s}$.
5. Sample $a_{n+1} \xleftarrow{\$} \mathbb{Z}_p^*$, and set $\alpha_{n+1} = a_{n+1} \cdot r^\Delta$.
6. Set $\mathsf{sk}_0 = (h, r, t, a_{n+1})$.
7. Set $\mathsf{pk}_0 = \left\{(c_0, c_1), (h_s)_{s \in S'}, \alpha_{n+1}\right\}$.
8. Output $(\mathsf{sk}_0, \mathsf{pk}_0)$.

$\mathsf{PCF.Gen}_1(1^\lambda)$:

1. Sample $\hat{\mathsf{k}} \xleftarrow{\$} \mathcal{K}$.
2. Compute $\mathsf{k} = (k_1, \dots, k_n) \leftarrow g(\hat{\mathsf{k}})$.
3. For $i \in [n]$, compute $\mathsf{com}_i = G^{s_i} \cdot H^{k_i} \pmod{N^2}$, where $s_i \xleftarrow{\$} \mathbb{Z}_N$.
   // Pedersen commitments of k_i.
4. Set $\mathsf{sk}_1 = (\mathsf{k}, (s_i)_{i \in [n]})$.
5. Set $\mathsf{pk}_1 = (\mathsf{com}_1, \cdots, \mathsf{com}_n)$.
6. Output $(\mathsf{sk}_1, \mathsf{pk}_1)$.

$\mathsf{PCF.KeyDer}(\sigma, \mathsf{sk}_\sigma, \mathsf{pk}_{1-\sigma})$:

1. If $\sigma = 0$:
   (a) Parse $\mathsf{sk}_0 = (h, r, t, a_{n+1})$.
   (b) Parse $\mathsf{pk}_1 = (\mathsf{com}_1, \cdots, \mathsf{com}_n)$.
   (c) For $i \in [n]$:
      i. compute $A_i = \mathsf{com}_i^t \pmod{N^2}$.
         $//A_i = G^{s_i \cdot t} \cdot H^{k_i \cdot t} \pmod{N^2}$.
      ii. compute $\hat{a}_i \leftarrow \mathsf{DDLog}_N(A_i)$.
         $//\hat{a}_i = (r' \cdot k_i)_0$.
      iii. set $a_i := g_q^{\hat{a}_i} \pmod p$.
         $//a_i = g_q^{(r' \cdot k_i)_0}$.
   (d) Set $\mathbf{a} = (a_1, \cdots, a_{n+1})$.
   (e) Set and output $k_0 = (h, \mathbf{a})$.
2. If $\sigma = 1$:
   (a) Parse $\mathsf{sk}_1 = (\mathsf{k}, (s_i)_{i \in [n]})$.
   (b) Parse $\mathsf{pk}_0 = \left\{(c_0, c_1), (h_s)_{s \in S'}, \alpha_{n+1}\right\}$.
   (c) For $i \in [n]$:
      i. compute $B_i = c_0^{s_i} \cdot c_1^{k_i} \pmod{N^2}$.
         $//B_i = G^{s_i \cdot t} \cdot H^{k_i \cdot t} \cdot (1+N)^{r' \cdot k_i}$.
      ii. compute $\hat{\alpha}_i = \mathsf{DDLog}_N(B_i)$.
         $//\hat{\alpha}_i = (r' \cdot k_i)_1$.
      iii. set $\alpha_i := g_q^{\hat{\alpha}_i} \pmod p$.
         $//\alpha_i = g_q^{(r' \cdot k_i)_1}$.
   (d) Set $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_{n+1})$.
   (e) Set and output $k_1 = (\boldsymbol{\alpha}, (h_s)_{s \in S'}, \mathsf{k}, \Delta)$.

$\mathsf{PCF.Eval}(1^\lambda, \sigma, k_\sigma, \hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_n))$:

1. If $\sigma = 0$:
   (a) Parse $k_0 = (h, \mathbf{a})$.
   (b) Compute $\mathbf{x} = (x_1, \ldots, x_n) \leftarrow f(\hat{\mathbf{x}})$.
   (c) Compute $r_0 = h^{\prod_{i=1}^{n} a_i^{x_i}}$.
   (d) Compute $r_1 = h^{\prod_{i=1}^{n} a_i^{x_i} \cdot a_{n+1}}$.
   (e) Set and output $y_0 \leftarrow (\mathsf{Hash}(r_0), \mathsf{Hash}(r_1))$.
2. If $\sigma = 1$:
   (a) Parse $k_1 = (\boldsymbol{\alpha}, (h_s)_{s \in S'}, \mathsf{k}, \Delta)$.
   (b) Compute $\mathbf{x} = (x_1, \ldots, x_n) \leftarrow f(\hat{\mathbf{x}})$.
   (c) Let $b = F_k(\hat{\mathbf{x}})$, and $s = \langle \mathbf{x}, \mathsf{k} \rangle - b \cdot \Delta$.
   (d) Compute $r = (h_s)^{\prod_{i=1}^{n} \alpha_i^{x_i} \cdot \alpha_{n+1}^{b}}$
   (e) Set and output $y_1 \leftarrow (b, \mathsf{Hash}(r))$.

Fig. 19. Public-Key PCF for OT Correlations from Paillier-ElGamal

Security Analysis

First we state the following lemma. We skip the proof and refer the reader to the construction idea explained in Section 6.3.

Lemma 6 (Correlated Evaluation Keys).
Let $\mathsf{PCF.Gen}$ *and* $\mathsf{PCF.KeyDer}$ *be the algorithms described in Figure 19. And let* $(\mathsf{sk}_\sigma, \mathsf{pk}_\sigma) \leftarrow \mathsf{PCF.Gen}(\sigma)$*, and* $k_\sigma \leftarrow \mathsf{PCF.KeyDer}(\sigma, \mathsf{sk}_\sigma, \mathsf{pk}_{1-\sigma})$ *for* $\sigma \in \{0, 1\}$*. It holds that* $k_0 = (h, \boldsymbol{a})$*,* $k_1 = (\boldsymbol{\alpha}, (h_s)_{s \in S'}, \mathsf{k}, \Delta)$*, where for a random element* $r \in \mathbb{Z}_p$*:*

- $h_s = h^{r^s}$ *for all* $s \in S'$*, and*
- $\boldsymbol{a} = (a_1, \ldots, a_{n+1})$ *and* $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{n+1})$ *such that* $\alpha_i = a_i \cdot r^{-k} \pmod p$ *for all* $i \in [n]$*, and* $\alpha_{n+1} = a_{n+1} \cdot r^{-\Delta}$.

We also state the following remark regarding the randomness space of operations over $\mathbb{NR}_{2N}$ in our construction:

Remark 4. The key generation and derivation phases of our PK-PCF protocol contain operations over the subgroup $\mathbb{NR}_{2N}$ that is of order $P'Q'$. However, since the two parties should perform their computations obliviously to the factorization of N, they sample their required randomness for computing commitments and encryptions from $\mathbb{Z}_N$ instead of $\mathbb{Z}_{P'Q'}$. This does not change the distribution of the resulting elements over $\mathbb{NR}_{2N}$, since

$$\Delta(\{r \pmod{P'Q'} : r \xleftarrow{\$} \mathbb{Z}_N\}, \mathcal{U}(\mathbb{Z}_{P'Q'})) = \frac{N \pmod{P'Q'}}{N} = \frac{2P' + 2Q' + 1}{4P'Q' + 2P' + 2Q' + 1} \leq \frac{1}{2^\lambda},$$

where $\Delta(\mathcal{D}_1, \mathcal{D}_2)$ denotes the statistical distance between the distributions $\mathcal{D}_1$ and $\mathcal{D}_2$.

We now prove the security of our PK-PCF construction.

Theorem 7 (PCF Security). *Assuming the hardness of sparse power-DDH (Assumption 2) and DCR (Assumption 3), the construction provided in Figure 19 is a secure public-key pseudorandom correlated function for OT correlations.*

Proof. We prove that our construction satisfies the properties of a PK-PCF.
Pseudorandom OT-Correlated Outputs. *We consider the following sequence of hybrid games:*

Hybrid $\mathcal{H}_0$*: This is the game* $\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},N,1}(\lambda)$*, where the view of an adversary consists of* $(1^\lambda, \mathsf{pk}_0, \mathsf{pk}_1, (x^{(i)}, y_0^{(i)}, y_1^{(i)})_{i \in [N(\lambda)]})$*, where each tuple* $(x^{(i)}, y_0^{(i)}, y_1^{(i)})$ *is generated by running the PCF evaluation algorithms of both parties on* $x^{(i)}$ *for* $i \in [N(\lambda)]$.

Hybrid $\mathcal{H}_1$*: In this hybrid, we replace the key generation and key derivation steps of Hybrid* $\mathcal{H}_0$ *by the following:*

1. *Sample* $h \xleftarrow{\$} \mathbb{G}$.
2. *Sample* $r \xleftarrow{\$} \mathbb{Z}_p$. // $\mathsf{PCF.Gen}_0$*:* $r = g_q^{r'}$*, where* $r' \xleftarrow{\$} \mathbb{Z}_q$.
   → *r has the same distribution in both cases.*
3. *For* $s \in S'$*, compute and set* $h_s := h^{r^s}$.
4. *Compute* $(c_0, c_1) = (G^t, H^t \cdot (1+N)^0)$*, where* $t \xleftarrow{\$} \mathbb{Z}_N$.
   // $\mathsf{PCF.Gen}_0$*:* $(c_0, c_1) = (G^t, H^t \cdot (1+N)^{r'})$.
   → *due to the semantic security of Paillier-ElGamal under the DCR assumption.*
5. *Sample* $a_{n+1} \xleftarrow{\$} \mathbb{Z}_p^*$*, and set* $\alpha_{n+1} = a_{n+1} \cdot r^{-\Delta}$.
6. *Set* $\mathsf{pk}_0 = \left\{(c_0, c_1), (h_s)_{s \in S'}, \alpha_{n+1}\right\}$.

7. *For* $i \in [n]$*, compute* $\mathsf{com}_i = G^{s_i} \cdot H^0 \pmod{N^2}$.
   // $\mathsf{PCF.Gen}_1$*:* $\mathsf{com}_i = G^{s_i} \cdot H^{\mathsf{k}_i}$.
   → *due to the perfectly hiding property of Pedersen commitments over* $\mathbb{NR}_{2N}$.
8. *Set* $\mathsf{pk}_1 = (\mathsf{com}_1, \ldots, \mathsf{com}_n)$.

// Instead of running $\mathsf{PCF.KeyDer}(\sigma, \mathsf{sk}_\sigma, \mathsf{pk}_{1-\sigma})$ *for* $\sigma = 0, 1$*:*

9. *Sample* $(a_1, \ldots, a_n) \xleftarrow{\$} \mathbb{Z}_p^n$*, and set* $\boldsymbol{a} = (a_1, \ldots, a_{n+1})$.
10. *Sample* $\hat{\mathsf{k}} \xleftarrow{\$} \mathcal{K}$*, and compute* $\mathsf{k} \leftarrow g(\hat{\mathsf{k}})$.
11. *For* $i \in [n]$*, compute and set* $\alpha_i = a_i \cdot r^{-\mathsf{k}_i}$.
12. *Set* $k_0 = (h, \boldsymbol{a})$*, and* $k_1 = (\boldsymbol{\alpha}, (h_s)_{s \in S}, \mathsf{k}, \Delta)$.
    → *due to the correctness of the key derivation (see Lemma 6).*

Hybrid $\mathcal{H}_1$ *differs from Hybrid* $\mathcal{H}_0$ *on the steps that are commented. Due to the highlighted arguments written for each step,* $\mathcal{H}_1$ *remains indistinguishable to* $\mathcal{H}_0$*. Note that in this hybrid, the view of the adversary contains simulated public keys* pk_0 *and* pk_1 *that are independent of any secrets.*

Hybrid $\mathcal{H}_2$*: This is the same as Hybrid* $\mathcal{H}_1$ *except that here, we replace*

$$\text{For } \sigma = 0, 1: \ y_\sigma \leftarrow \mathsf{PCF.Eval}(\sigma, k_\sigma, x^{(i)}),$$

with

$$(y_0^{(i)}, y_1^{(i)}) \xleftarrow{\$} \mathsf{OT}(1^\lambda).$$

Regarding the security of the Naor-Reingold CPRF and Theorems 4 and 5, hybrids $\mathcal{H}_2$ *and* $\mathcal{H}_1$ *are indistinguishable.*

Hybrid $\mathcal{H}_3$*: This is the same as Hybrid* $\mathcal{H}_2$ *except that here, we undo the changes from hybrid* $\mathcal{H}_1$ *and replace the whole batch of modified steps with*

$$\text{For } \sigma = 0, 1 : (\mathsf{sk}_\sigma, \mathsf{pk}_\sigma) \leftarrow \mathsf{PCF.Gen}(1^\lambda, \mathsf{pp}, \sigma).$$

This Hybrid remains indistinguishable to Hybrid $\mathcal{H}_2$ *due to the same reasons explained in Hybrid* $\mathcal{H}_1$ *for the steps that simulated the two public keys. The evaluation keys are not used in* $\mathcal{H}_2$ *anymore, and therefore, it does not change the view of the adversary that we do not generate them in this hybrid.*

Note that hybrid $\mathcal{H}_3$ *has the same distribution as* $\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},N,0}(\lambda)$*. This concludes the proof of pseudorandom OT-correlated output of the construction.*

Security. *We prove the security for each* $\sigma \in \{0, 1\}$.

- *For* $\sigma = 0$*, consider the following sequence of hybrid games:*

  Hybrid $\mathcal{H}_0$*: This is the game* $\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},N,\sigma=0,0}(\lambda)$*, where the view of party* 0 *considered as the adversary consists of*

  $$(1^\lambda, \mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_0, (x^{(i)}, y_0^{(i)}, y_1^{(i)})_{i \in [N(\lambda)]}),$$

  where $y_1^{(i)} \leftarrow \mathsf{PCF.Eval}(1, k_1, x^{(i)})$ *for all* $i \in [N(\lambda)]$.

Hybrid $\mathcal{H}_1$*: This is the same as hybrid* $\mathcal{H}_0$*, but here, we remove the steps*

$$(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{PCF.Gen}(1), \quad \textit{and} \quad k_1 \leftarrow \mathsf{PCF.KeyDer}(1, \mathsf{sk}_1, \mathsf{pk}_0),$$

and generate pk_1 *and* k_1 *as follows:*

1. *Let* $(\mathsf{sk}_0, \mathsf{pk}_0) \leftarrow \mathsf{PCF.Gen}(0)$.
2. *Parse* $\mathsf{sk}_0 = (h, r, t, a_{n+1})$*, and* $\mathsf{pk}_0 = \{(c_0, c_1), (h^{r^s})_{s \in S'}, \alpha_{n+1}\}$.

   // To generate pk_1*:*
3. *For* $i \in [n]$*:*
   (a) *Sample* $s_i \xleftarrow{\$} \mathbb{Z}_N$.
   (b) *Compute* $\mathsf{com}_i = G^{s_i}$.
4. *Output* $\mathsf{pk}_1 = (\mathsf{com}_1, \ldots, \mathsf{com}_n)$.

   // To generate k_1*:*
5. *Run* $k_0 \leftarrow \mathsf{PCF.KeyDer}(0, \mathsf{sk}_0, \mathsf{pk}_1)$.
6. *parse* $k_0 = (h, \boldsymbol{a})$.
7. *Sample* $\hat{\mathsf{k}} \xleftarrow{\$} \mathcal{K}$*, and compute* $\mathsf{k} = g(\hat{\mathsf{k}})$.
8. *For* $i \in [n]$*, compute and set* $\alpha_i = a_i \cdot r^{-k_i} \pmod p$.
9. *Let* $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{n+1})$.
10. *Set* $k_1 = (\boldsymbol{\alpha}, (h_s)_{s \in S'}, \mathsf{k}, \Delta)$.

Here, differently from Hybrid $\mathcal{H}_0$*, the public key* pk_1 *contains random commitments that are independent of the evaluation key* k_1*. Consequently, we set the key* k_1 *with respect to* sk_0 *such that for each* $i \in [N(\lambda)]$*, the OT correlation between* $y_0^{(i)}$ *and* $y_1^{(i)}$ *remains correct.*
The public key $\mathsf{pk}_1 = (\mathsf{com}_1, \ldots, \mathsf{com}_n)$ *retains the same distribution as in* $\mathcal{H}_0$*, since Pedersen commitments are perfectly hiding over* $\mathbb{NR}_{2N}$*. As a result, Hybrid* $\mathcal{H}_1$ *remains indistinguishable from Hybrid* $\mathcal{H}_0$.

Hybrid $\mathcal{H}_2$*: This is the same as Hybrid* $\mathcal{H}_1$ *except that here, we compute* $k_0 \leftarrow \mathsf{PCF.KeyDer}(0, \mathsf{sk}_0, \mathsf{pk}_1)$*, and then for each* $i \in [N(\lambda)]$*, we replace*

$$y_1^{(i)} \leftarrow \mathsf{PCF.Eval}(1, k_1, x^{(i)})$$

in $\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},N,\sigma=0,0}(\lambda)$ *with*

$$y_0^{(i)} \leftarrow \mathsf{PCF.Eval}(0, k_0, x^{(i)}), \quad \textit{and} \quad y_1^{(i)} \leftarrow \mathsf{RSample}(1^\lambda, 0, y_0^{(i)}).$$

Regarding the correctness of the key derivation process, the security of the Naor-Reingold CPRF and Theorem 4, hybrids $\mathcal{H}_2$ *and* $\mathcal{H}_1$ *are indistinguishable.*

Hybrid $\mathcal{H}_3$*: This is the same as hybrid* $\mathcal{H}_2$ *except that here, we undo the changes that we made in hybrid* $\mathcal{H}_1$ *and bring back the algorithm* $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{PCF.Gen}(1^\lambda, \mathsf{pp}, 1)$*. The evaluation key* k_1 *is not used in hybrid* $\mathcal{H}_2$ *anymore, and therefore, removing it does not change the view of party 0. This hybrid remains indistinguishable to hybrid* $\mathcal{H}_2$ *for the same arguments explained in hybrid* $\mathcal{H}_1$.

Note that hybrid $\mathcal{H}_3$ *has the same distribution as* $\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},N,\sigma=0,1}(\lambda)$*. This concludes the proof of security of our PK-PCF construction against party* 0.

- *For* $\sigma = 1$*, consider the following sequence of hybrid games:*

  Hybrid $\mathcal{H}_0$*: This is the game* $\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},N,\sigma=1,0}(\lambda)$.

  Hybrid $\mathcal{H}_1$*: This is the same as hybrid* $\mathcal{H}_0$*, but here, we remove the steps*

  $$(\mathsf{sk}_0, \mathsf{pk}_0) \leftarrow \mathsf{PCF.Gen}(0), \quad \textit{and} \quad k_0 \leftarrow \mathsf{PCF.KeyDer}(0, \mathsf{sk}_0, \mathsf{pk}_1),$$

  and generate pk_0 *and* k_0 *as explained in the following. We also have to determine a part of the randomness of the* $\mathsf{PCF.Gen}(1)$ *to generate a consistent public key* pk_1.

1. *Let* $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{PCF.Gen}(1)$.
2. *Parse* $\mathsf{sk}_1 = (\mathsf{k}, (s_i)_{i\in[n]})$, *and* $\mathsf{pk}_1 = (\mathsf{com}_1, \ldots, \mathsf{com}_n)$.

// To generate pk_0*:*

3. *Sample* $h \xleftarrow{\$} \mathbb{G}$.
4. *Sample* $r \xleftarrow{\$} \mathbb{Z}_p$.
5. *For* $s \in S'$, *compute and set* $h_s := h^{r^s}$.
6. *Compute* $(c_0, c_1) = (G^t, H^t)$, *where* $t \xleftarrow{\$} \mathbb{Z}_N$.
7. *Sample* $\alpha_{n+1} \xleftarrow{\$} \mathbb{Z}_p^*$.
8. *Set* $\mathsf{pk}_0 = ((c_0, c_1), (h_s)_{s\in S'}, \alpha_{n+1})$.

// To generate k_0*:*

9. *Run* $k_1 \leftarrow \mathsf{PCF.KeyDer}(1, \mathsf{sk}_1, \mathsf{pk}_0)$.
10. *Parse* $k_1 = (\boldsymbol{\alpha}, (h_s)_{s\in S'}, \mathsf{k}, \Delta)$.
11. *For* $i \in [n+1]$, *compute and set* $a_i = \alpha_i \cdot r^{k_i} \pmod p$.
12. *Let* $\boldsymbol{a} = (a_1, \ldots, a_{n+1})$.
13. *Set* $\mathsf{k}_0 = (h, \boldsymbol{a})$.

Here, differently from Hybrid $\mathcal{H}_0$*, the public key* pk_0 *contains ciphertexts that are independent from the evaluation key* k_0*. Consequently, we set the key* k_0 *with respect to* sk_1 *such that for each* $i \in [N(\lambda)]$*, the OT correlation between* $y_0^{(i)}$ *and* $y_1^{(i)}$ *remains correct.*
The public key $\mathsf{pk}_0 = ((c_0, c_1), (h_s)_{s\in S'}, \alpha_{n+1})$ *retains the same distribution as in* $\mathcal{H}_0$*, since Paillier-ElGamal encryption is semantically secure over* $\mathbb{NR}_{2N}$*. As a result, Hybrid* $\mathcal{H}_1$ *remains indistinguishable from Hybrid* $\mathcal{H}_0$*.*

Hybrid $\mathcal{H}_2$*: This is the same as Hybrid* $\mathcal{H}_1$ *except that here, we compute* $k_1 \leftarrow \mathsf{PCF.KeyDer}(0, \mathsf{sk}_1, \mathsf{pk}_0)$*, and then replace each*

$$y_0^{(i)} \leftarrow \mathsf{PCF.Eval}(0, k_0, x^{(i)})$$

in $\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},N,\sigma=1,0}(\lambda)$ *with*

$$y_1^{(i)} \leftarrow \mathsf{PCF.Eval}(1, k_1, x^{(i)}), \text{ and } y_0^{(i)} \leftarrow \mathsf{RSample}(1^\lambda, 1, y_1^{(i)}).$$

Regarding the security of the Naor-Reingold CPRF and Theorem 4, hybrids $\mathcal{H}_2$ *and* $\mathcal{H}_1$ *are indistinguishable.*

Hybrid $\mathcal{H}_3$*: This is the same as hybrid* $\mathcal{H}_2$ *except that here, we undo the changes that we made in hybrid* $\mathcal{H}_1$ *and bring back the algorithms*

$$(\mathsf{sk}_0, \mathsf{pk}_0) \leftarrow \mathsf{PCF.Gen}(0), \text{ and } (\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{PCF.Gen}(1).$$

The evaluation key k_0 *is not used in hybrid* $\mathcal{H}_2$ *anymore, and therefore, removing it does not change the view of the adversary (party 1). This hybrid remains indistinguishable to hybrid* $\mathcal{H}_2$ *for the same arguments explained in hybrid* $\mathcal{H}_1$*.*

Note that hybrid $\mathcal{H}_3$ *has the same distribution as* $\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},N,\sigma=1,1}(\lambda)$*. This concludes the proof of security of our PK-PCF construction against party* 1*.*

□

Remark 5 (security in the case of semi-honest parties). When proving the security of the PK-PCF scheme presented in Figure 19 against either of the two parties (proof of Theorem 7), we perform steps that are independent from the randomness used in the key generation algorithm of that party. Therefore, the same proof holds against a semi-honest party that can determine the randomness of the key generation algorithm.

6.4 Reducing The Public Keys Size to $\mathcal{O}(n^{2/3})$

In the public-key PCF of Figure 19, the public keys are of the form

$$\mathsf{pk}_0 = \left\{(c_0, c_1), (h_s)_{s\in S'}, \alpha_{n+1}\right\}, \text{ and } \mathsf{pk}_1 = (\mathsf{com}_1, \cdots, \mathsf{com}_n),$$

where (c_0, c_1) is a Paillier-ElGamal ciphertext and $(\mathsf{com}_1, \cdots, \mathsf{com}_n)$ are n Pedersen commitments over $\mathbb{NR}_{2N}$. Regarding the key sizes, pk_0 contains 2 elements of $\mathbb{NR}_{2N}$, while pk_1 contains n such elements. In this section we aim to find a better balance for the size of pk_0 and pk_1.
The key observation is that pk_1 that includes a list of Pedersen commitments can be easily made compact using generalized Pedersen commitments which allow committing to n different values by generating a single commitment. However, this must be done while maintaining the correct correlation of derived PCF keys for both parties. In the following, we explain how we achieve the correctness by including more Paillier-ElGamal ciphertexts in pk_0 while reducing the total size of the public keys.
Let $0 < m \leq n$ be a block size. The idea is the following:

- Let $G, H_1, \ldots, H_m$ be random elements of $\mathbb{NR}_{2N}$.
- Party 1 divides the wPRF key k into consecutive subvectors $\mathsf{k}_1, \ldots, \mathsf{k}_\delta$, each of length m. It then commits to each subvector k_u by generating generalized Pedersen commitments $\mathsf{com}_u = G^{s_u} \cdot \prod_{j=1}^{m} H_j^{k_u^{(j)}}$, for all $u \in [\delta]$.
- Party 0 generates m^2 Paillier-ElGamal encryptions of r' with randomness reuse as follows:

$$\begin{aligned}
(c_1^0, c_1^1, \ldots, c_1^m) &= (G^{t_1}, H_1^{t_1} \cdot (1+N)^{r'}, H_2^{t_1}, \ldots, H_m^{t_1}) \\
(c_2^0, c_2^1, \ldots, c_2^m) &= (G^{t_2}, H_1^{t_2}, H_2^{t_2} \cdot (1+N)^{r'}, \ldots, H_m^{t_2}) \\
&\vdots \\
(c_m^0, c_m^1, \ldots, c_m^m) &= (G^{t_m}, H_1^{t_m}, H_2^{t_m}, \ldots, H_m^{t_m} \cdot (1+N)^{r'}).
\end{aligned}$$

- Party 0 publishes the m^2 ciphertexts $(c_i^0, c_i^j)_{i,j\in[m]}$ as a part of pk_0 and party 1 publishes the n/m generalized commitments $\mathsf{com}_1, \ldots, \mathsf{com}_\delta$ as a part of pk_1.

By inspection, one can see that for each $u \in [\delta]$ and each $v \in [m]$, it holds that

$$\mathsf{com}_u^{t_v} \cdot (1+N)^{r' \cdot k_u^{(v)}} = (c_v^0)^{s_u} \cdot \prod_{j=1}^{m} (c_v^j)^{k_u^{(j)}}.$$

Therefore, the two parties can first derive multiplicative shares of $(1+N)^{r' \cdot k_u^{(v)}}$, and as before, after applying the DDLog algorithm over their shares and mapping the result in $\mathbb{G}$, they can compute their PCF keys.

Doing as explained above yields publishing $m^2 + 2m + n/m$ elements (including $G, H_1, \ldots, H_m$ elements of the public parameters), where n denotes the length of the wPRF key k. Minimizing with respect to m results in $\mathcal{O}(n^{2/3})$ elements.

The security analysis of the construction is similar to that of the public-key PCF presented in Figure 19 (See proof of Theorem 7) where we now leverage the perfectly-hiding property of generalized Pedersen commitments and semantic security of Paillier-ElGamal ciphertexts with randomness reuse over $\mathbb{NR}_{2N}$.

The construction of the resulting optimized public-key PCF is provided in Figure 20.

Optimized PK-PCF for OT Correlations from sparse power-DDH and DCR (with compressed public keys)

Requires:

- Let $F : \mathcal{K} \times \mathcal{X} \to \{0,1\}$ be an $\mathsf{IPM\text{-}wPRF}$ with respect to an IPM set S and predicate functions f, g (See Definition 13). We assume that S is Δ-antiperiodic, and consider $S' = S \cup \{-\Delta/2, \ldots, -1\}$.
- Let p be a safe prime, i.e., $p = 2q+1$ for a prime q, and $N = PQ$ for primes P and Q.
- Let DDLog_N be the distributed discrete logarithm algorithm as in Lemma 5.
- Let Hash be a hash function modeled as a random oracle.

$\mathsf{PCF.Setup}(1^\lambda)$:

1. $(\mathbb{G}, g, p) \xleftarrow{\$} \mathsf{GenPar}(1^\lambda)$.
2. Sample a generator g_q of QR_p.
3. Sample $G' \xleftarrow{\$} \mathbb{Z}_{N^2}$, and set $G \leftarrow (G')^{2N} \pmod{N^2}$.
4. For $i \in [m]$, sample $d_i \xleftarrow{\$} \mathbb{Z}_{N^2}$, and set $H_i \leftarrow G^{d_i} \pmod{N^2}$.
5. Output $\mathsf{pp} = (\mathbb{G}, p, g_q, (G, H_1, \ldots, H_m), F)$.

$\mathsf{PCF.Gen}_0(1^\lambda)$:

1. Sample $h \xleftarrow{\$} \mathbb{G}$.
2. Sample $r' \xleftarrow{\$} \mathbb{Z}_q$, and set $r := g_q^{r'} \pmod{p}$.
3. For $i \in [m]$ do:
   (a) Sample $t_i \xleftarrow{\$} \mathbb{Z}_N$.
   (b) Compute $\mathbf{c}_i = (c_i^0, c_i^1, \ldots, c_i^m)$, where $c_i^0 = G^{t_i}$, and $c_i^i = H_i^{t_i} \cdot (1+N)^{r'}$, and $c_i^j = H_j^{t_i}$ for all $j \neq i \in [m]$.
   // m^2 Paillier-ElGamal encryptions of r'.
4. For $s \in S'$, compute and set $h_s := h^{r^s}$.
5. Sample $a_{n+1} \xleftarrow{\$} \mathbb{Z}_p^*$, and set $\alpha_{n+1} = a_{n+1} \cdot r^\Delta$.
6. Set $\mathsf{sk}_0 = (h, r, (t_i)_{i\in[m]}, a_{n+1})$.
7. Set $\mathsf{pk}_0 = \left\{(c_i^0, (c_i^j))_{i,j\in[m]}, (h_s)_{s\in S'}, \alpha_{n+1}\right\}$.
8. Output $(\mathsf{sk}_0, \mathsf{pk}_0)$.

$\mathsf{PCF.Gen}_1(1^\lambda)$:

1. Sample $\hat{\mathsf{k}} \xleftarrow{\$} \mathcal{K}$.
2. Compute $\mathsf{k} \leftarrow g(\hat{\mathsf{k}})$.
3. Partition k into δ subvectors $\mathsf{k}_1, \mathsf{k}_2, \ldots, \mathsf{k}_\delta$ of length m.
4. For $u \in [\delta]$:
   (a) Parse $\mathsf{k}_u = (k_u^{(1)}, \ldots, k_u^{(m)})$.
   (b) Compute $\mathsf{com}_u = G^{s_u} \cdot \prod_{j=1}^{m} H_j^{k_u^{(j)}}$, where $s_u \xleftarrow{\$} \mathbb{Z}_N$.
   // Generalized Pedersen commitment of k_u.
5. Set $\mathsf{sk}_1 = ((\mathsf{k}_u)_{u\in[\delta]}, (s_u)_{u\in[\delta]})$.
6. Set $\mathsf{pk}_1 = (\mathsf{com}_1, \ldots, \mathsf{com}_\delta)$.
7. Output $(\mathsf{sk}_1, \mathsf{pk}_1)$.

$\mathsf{PCF.KeyDer}(\sigma, \mathsf{sk}_\sigma, \mathsf{pk}_{1-\sigma})$:

1. If $\sigma = 0$:
   (a) Parse $\mathsf{sk}_0 = (h, r, (t_i)_{i\in[m]}, a_{n+1})$.
   (b) Parse $\mathsf{pk}_1 = (\mathsf{com}_1, \cdots, \mathsf{com}_\delta)$.
   (c) For $u \in [\delta]$, and $v \in [m]$: set $A_i = (\mathsf{com}_u)^{t_v}$, where $i = m(u-1)+v$.
   $//A_i = G^{t_v \cdot s_u} \cdot \prod_{j=1}^{m} H_j^{k_u^{(j)} \cdot t_v}$.
   (d) For $i \in [n]$:
      i. compute $\hat{a}_i \leftarrow \mathsf{DDLog}_N(A_i)$.
      ii. set $a_i := g_q^{\hat{a}_i}$.
   (e) Set $\mathbf{a} = (a_1, \cdots, a_{n+1})$.
   (f) Set and output $k_0 = (h, \mathbf{a})$.
2. If $\sigma = 1$:
   (a) Parse $\mathsf{sk}_1 = ((\mathsf{k}_u)_{u\in[\delta]}, (s_u)_{u\in[\delta]})$.
   (b) Parse $\mathsf{pk}_0 = \left\{(c_i^0, (c_i^j))_{i,j\in[m]}, (h_s)_{s\in S'}, \alpha_{n+1}\right\}$.
   (c) For $u \in [\delta]$, and $v \in [m]$: set $B_i = (c_v^0)^{s_u} \cdot \prod_{j=1}^{m} (c_v^j)^{k_u^{(j)}}$, where $i = m(u-1)+v$.
   $//B_i = G^{t_v \cdot s_u} \cdot \prod_{j=1}^{m} H_j^{k_u^{(j)} \cdot t_v} \cdot (1+N)^{r' \cdot k_u^{(v)}}$.
   (d) For $i \in [n]$:
      i. compute $\hat{\alpha}_i \leftarrow \mathsf{DDLog}_N(B_i)$.
      ii. set $\alpha_i = g_q^{\hat{\alpha}_i} \pmod{p}$.
   (e) Set $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{n+1})$.
   (f) Set and output $k_1 = (\boldsymbol{\alpha}, (h_s)_{s\in S'}, \mathsf{k}, \Delta)$.

$\mathsf{PCF.Eval}(1^\lambda, \sigma, k_\sigma, \hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_n))$:

1. If $\sigma = 0$:
   (a) Parse $k_0 = (h, \mathbf{a})$.
   (b) Compute $\mathbf{x} = (x_1, \ldots, x_n) \leftarrow f(\hat{\mathbf{x}})$.
   (c) Compute $r_0 = h^{\prod_{i=1}^{n} a_i^{x_i}}$.
   (d) Compute $r_1 = h^{\prod_{i=1}^{n} a_i^{x_i} \cdot a_{n+1}}$.
   (e) Set and output $y_0 \leftarrow (\mathsf{Hash}(r_0), \mathsf{Hash}(r_1))$.
2. If $\sigma = 1$:
   (a) Parse $k_1 = (\boldsymbol{\alpha}, (h_s)_{s \in S'}, \mathsf{k}, \Delta)$.
   (b) Compute $\mathbf{x} = (x_1, \ldots, x_n) \leftarrow f(\hat{\mathbf{x}})$.
   (c) Let $b = F_{\mathsf{k}}(\hat{\mathbf{x}})$, and $s = \langle \mathbf{x}, \mathsf{k} \rangle - b \cdot \Delta$.
   (d) Compute $r = (h_s)^{\prod_{i=1}^{n} \alpha_i^{x_i} \cdot \alpha_{n+1}^{b}}$.
   (e) Set and output $y_1 \leftarrow (b, \mathsf{Hash}(r))$.

Fig. 20. Public-Key PCF for OT Correlations with Public Key Size $\mathcal{O}(n^{2/3})$.

7 DV-NIZKs from PK-PCFs

In this section, taking the advantage of *non-interactive* PK-PCF, we propose a new construction of *reusable* DV-NIZK argument of knowledge from a compiler that combines a sigma protocol for general NP language and a public-key PCF. Specifically, our *reusable* DV-NIZK comes from three ingredients:

- A Σ-protocol [CDS94] with 1-bit challenges for an NP-complete language $\mathcal{L}$, for example Blum's protocol for graph Hamiltonicity [Blu86].
- A *strong* public key PCF for OT correlation where the key evaluation of each party can be *silently* obtained from their own secret key and public key of the other.
- A *non-reusable* DV-NIZK with computational adaptive soundness knowledge and adaptive zero-knowledge properties.

We also show how to enhance our construction to obtain a *reusable* DV-NIZK based on a *weak* public key PCF instead of the strong one. We state the formal result below.

Theorem 8. *If there exists a weak public-key PCF then we can construct a reusable DV-NIZK argument of knowledge for* NP *language from a Σ-protocol and a non-reusable DV-NIZK scheme.*

7.1 Construction of reusable DVNIZK

***Σ*-protocol.** A Σ-protocol is a 3-move, interactive proof (fig. 21) which consists of three algorithms $(\Sigma.P_1, \Sigma.P_2, \Sigma.V)$. While $(x, w) \in \mathcal{R}$, the prover inputs (x, w) to $\Sigma.P_1$ and produces a random message a to the verifier while a is considered as a commitment of x. The verifier then samples a random challenge bit b and sends it to the prover. The prover runs the algorithm $\Sigma.P_2(x, w, a, b)$ to produce a response z_b corresponding to bit challenge b and sends z_b to the verifier. Finally, the verifier runs $\Sigma.V(x, a, b, z_b)$ to decide whether accept or reject the proof.

Σ-protocol satisfies 3 properties: perfect completeness, 2-special soundness (given two accepting transcripts $(a, 0, z_0), (a, 1, z_1)$ of a statement x then there exists an efficient algorithm $\Sigma.\mathsf{Ext}(x, a, z_0, z_1)$ to extract the witness w) and special honest-verifier zero-knowledge (given b ahead of time, there exists a simulator $\mathsf{Sim}(x, b)$ simulating the transcript (a, b, z_b) without knowing a witness). The Blum's Hamiltonicity protocol [Blu86] is an instantiation of Σ-protocol with 1-bit challenges for NP language.

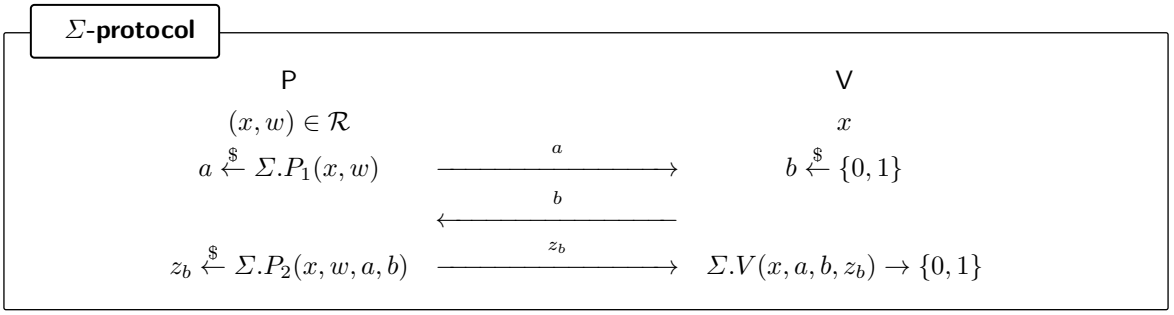

Fig. 21. Σ-protocol with challenge space {0,1}.

Public-key PCF. To apply PK-PCF fig. 10 to our reusable DVNIZK scheme, we provide a higher security definition of PK-PCF compared to the one in fig. 17. The formal definition is shown as below:

Strong Security. Define $\mathsf{Supp}(\mathsf{PCF.Gen}(1^\lambda, \mathsf{pp}, \sigma))$ be a constraint relation such that for all $(\mathsf{sk}_\sigma, \mathsf{pk}_\sigma) \in \mathsf{Supp}(\mathsf{PCF.Gen}(1^\lambda, \mathsf{pp}, \sigma))$ then there exists a public coin $\rho \in \mathsf{pp}$ such that $(\mathsf{sk}_\sigma, \mathsf{pk}_\sigma) = \mathsf{PCF.Gen}(1^\lambda, \mathsf{pp}, \sigma, \rho)$.

Experiment PCF Strong Security

$\underline{\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},N,\sigma,0}(\lambda)}$:

$\mathsf{pp} \leftarrow \mathsf{PCF.Setup}(1^\lambda)$
$(\mathsf{sk}_\sigma, \mathsf{pk}_\sigma) \leftarrow \mathsf{PCF.Gen}(1^\lambda, \mathsf{pp}, \sigma)$
$(\mathsf{sk}_{1-\sigma}, \mathsf{pk}_{1-\sigma}) \leftarrow \mathsf{PCF.Gen}(1^\lambda, \mathsf{pp}, 1-\sigma)$
$k_{1-\sigma} \leftarrow \mathsf{PCF.KeyDer}(1-\sigma, \mathsf{sk}_{1-\sigma}, \mathsf{pk}_\sigma)$
For $i = 1, \ldots, N(\lambda)$:
  $x^{(i)} \xleftarrow{\$} \{0,1\}^{n(\lambda)}$
  $y^{(i)}_{1-\sigma} \leftarrow \mathsf{PCF.Eval}(1-\sigma, k_{1-\sigma}, x^{(i)})$
For all $i \in [N(\lambda)]$:
$b \leftarrow \mathcal{A}(1^\lambda, \mathsf{pk}_0, \mathsf{pk}_1, \sigma, \mathsf{sk}_\sigma, (x^{(i)}, y^{(i)}_{1-\sigma}))$
Output b

$\underline{\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},N,\sigma,1}(\lambda)}$:

$\mathsf{pp} \leftarrow \mathsf{PCF.Setup}(1^\lambda)$
$\forall \mathsf{pk}_\sigma$ such that: $\exists \mathsf{sk}_\sigma$,
$(\mathsf{sk}_\sigma, \mathsf{pk}_\sigma) \in \mathsf{Supp}(\mathsf{PCF.Gen}(1^\lambda, \mathsf{pp}, \sigma))$
$(\mathsf{sk}_{1-\sigma}, \mathsf{pk}_{1-\sigma}) \leftarrow \mathsf{PCF.Gen}(1^\lambda, \mathsf{pp}, 1-\sigma)$
$k_\sigma \leftarrow \mathsf{PCF.KeyDer}(\sigma, \mathsf{sk}_\sigma, \mathsf{pk}_{1-\sigma})$
For $i = 1, \ldots, N(\lambda)$:
  $x^{(i)} \xleftarrow{\$} \{0,1\}^{n(\lambda)}$
  $y^{(i)}_\sigma \leftarrow \mathsf{PCF.Eval}(\sigma, k_\sigma, x^{(i)})$
  $y^{(i)}_{1-\sigma} \leftarrow \mathsf{RSample}(1^\lambda, \sigma, y^{(i)}_\sigma)$
For all $i \in [N(\lambda)]$:
$b \leftarrow \mathcal{A}(1^\lambda, \mathsf{pk}_0, \mathsf{pk}_1, \sigma, \mathsf{sk}_\sigma, (x^{(i)}, y^{(i)}_{1-\sigma}))$
Output b

Fig. 22. Strong security of a PK-PCF. Here, RSample is the algorithm for reverse sampling $\mathcal{Y}$ as in the definition 3.

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.

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

where $\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},N,\sigma,b}$ ($b \in \{0,1\}$) is defined as in Figure 22. In particular, the adversary is given access to $N(\lambda)$ samples.

Non-reusable DV-NIZK arguments. Let $\mathcal{L}_P$ be an NP language associated with an NP relation $\mathcal{R}_\mathcal{P}$. We take advantage of a (one-time) DV-NIZK scheme for $\mathcal{L}_P$ which consists of three algorithms $\mathsf{dv} = (\mathsf{dv.Setup}, \mathsf{dv.P}, \mathsf{dv.V})$. For convenience, we define the notion of a DV-NIZK as below.

– $\mathsf{dv.Setup}(1^\lambda) \to (\mathsf{dv.crs}, \mathsf{dv}.\mathcal{T})$.

– $\mathsf{dv.P}(\mathsf{dv.crs}, x, w) \to \mathsf{dv}.\pi$.
– $\mathsf{dv.V}(\mathsf{dv.crs}, x, \mathsf{dv}.\pi, \mathsf{dv}.\mathcal{T}) \to \{0,1\}$.

The dv scheme needs to satisfy perfect completeness, computational (bounded) adaptive knowledge soundness and computational zero-knowledge properties. The non-reusable DV-NIZK can be constructed from a public-key encryption scheme [PsV06] by λ invocations of Σ-protocol. In our construction, dv is used to prove that the prover uses the correct form of pk_P to generate the proof. Specifically, for a statement pk_P and a witness sk_P we consider $\mathsf{pk}_P \in \mathcal{L}_p$, $(\mathsf{pk}_P, \mathsf{sk}_P) \in \mathcal{R}_P$ if $(\mathsf{pk}_P, \mathsf{sk}_P) \in \mathsf{Supp}(\mathsf{PCF.Gen}_0(1^\lambda, \mathsf{PCF.pp}))$.

Reusable DV-NIZK from PK-PCF. We show how to combine a Σ-protocol (fig. 21), a *non-reusable* DV-NIZK, and a *strong* public-key PCF to obtain a *reusable* DV-NIZK scheme for all NP language. Let $\mathcal{R}$ be an NP language, $(x, w) \in \mathcal{R}$, then our re-usable DV-NIZK scheme $(\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ is defined as detailed in Figure 23.

Reusable DV-NIZK: $\Pi(\mathsf{Setup}, \mathsf{P}, \mathsf{V})$

- $\mathsf{Setup}(1^\lambda)$.
  - Compute $\mathsf{PCF.pp} \leftarrow \mathsf{PCF.Setup}(1^\lambda)$, $(\mathsf{pk}_V, \mathsf{sk}_V) \leftarrow \mathsf{PCF.Gen}_1(1^\lambda, \mathsf{PCF.pp})$ and $(\mathsf{dv.crs}, \mathsf{dv}.\mathcal{T}) \leftarrow \mathsf{dv.Setup}(1^\lambda)$.
  - Return $\mathsf{crs} := (\mathsf{PCF.pp}, \mathsf{dv.crs}, \mathsf{pk}_V)$ and $\mathcal{T} := (\mathsf{sk}_V, \mathsf{dv}.\mathcal{T})$.
- $\mathsf{P}(\mathsf{crs}, x, w)$.
  1. Compute $(\mathsf{pk}_P, \mathsf{sk}_P) \leftarrow \mathsf{PCF.Gen}_0(1^\lambda, \mathsf{PCF.pp})$, $k_0 \leftarrow \mathsf{PCF.KeyDer}(0, \mathsf{sk}_P, \mathsf{pk}_V)$ and generate $\mathsf{dv}.\pi = \mathsf{dv.P}(\mathsf{dv.crs}, \mathsf{pk}_P, \mathsf{sk}_P)$.
  2. For each $i \in [1, \lambda]$ generate all Σ-protocol transcripts for both challenges $b \in [0,1]$:
$$a_i = \Sigma.P_1(x, w)$$
$$z_{i,b} = \Sigma.P_2(x, w, a_i, j) \text{ for } b \in [0,1]$$
  3. Compute $(r_{i,0}, r_{i,1}) = \mathsf{PCF.Eval}(1^\lambda, 0, k_0, x)$ and for each $i \in [1, \lambda]$, $b \in [0,1]$ define $m_{i,b} = z_{i,b} \oplus r_{i,b}$.
  4. Define the output $\pi := (\mathsf{pk}_P, \mathsf{dv}.\pi, \{a_i, m_{i,0}, m_{i,1}\}_{i \le \lambda})$.
- $\mathsf{V}(\mathsf{crs}, x, \pi, \mathcal{T})$.
  1. Check if $\mathsf{dv.V}(\mathsf{dv.crs}, \mathsf{pk}_P, \mathsf{dv}.\pi, \mathsf{dv}.\mathcal{T}) = 1$ then
     Compute $k_1 \leftarrow \mathsf{PCF.KeyDer}(1, \mathsf{sk}_V, \mathsf{pk}_P)$ and for each $i \in [1, \lambda]$, compute as follows:
$$\mathsf{PCF.Eval}(1^\lambda, 1, k_1, x) = (b_i, r_{i,b_i}), \quad z_{i,b_i} = m_{i,b_i} \oplus r_{i,b_i}$$
     The output is defined $\mathsf{V}(\mathsf{crs}, x, \pi, \mathcal{T}) = 1$ if $\Sigma.V(x, a_i, b_i, z_{i,b_i}) = 1$ for all $i \in [1, \lambda]$.
  2. Otherwise, $\mathsf{V}(\mathsf{crs}, x, \pi, \mathcal{T}) \to 0$.

Fig. 23. Reusable DV-NIZK construction.

The reusable knowledge soundness of our DV-NIZK construction (fig. 23) follows from the special soundness of Σ-protocol, knowledge soundness of DV-NIZK to extract a valid witness with high probability whenever prover outputs an accepted proof. In particular, we build a simulator can simulate the verification query without knowing the sk_V and an efficient extractor Ext can extract a valid witness by using dv.Ext to get the sk_P then later use Σ.Ext with accepted transcripts to extract witness w.

Our DV-NIZK scheme (fig. 23) is zero-knowledge because there exists a simulator knowing ahead of time the challenge $b \in [0,1]$ that can simulate the view of the verifier $(a_i, b, z_{i,b})$ without knowing the witness. Therefore, the view of the verifier in Π can be simulated by a simulator that for each i-invocation of the proof defines $m_{i,b} := z_{i,b} \oplus r_{i,b}$ and picks a dummy item for $m_{i,1-b}$.

Theorem 9 (Completeness). *If* dv, Σ *are complete and* PCF *is correct then DV-NIZK scheme from fig. 23 is complete.*

Proof. Consider $(x, w) \in \mathcal{R}$, *by completeness of the non-reusable DV-NIZK,* $\mathsf{dv.V}(\mathsf{dv.crs}, \mathsf{pk}_P, \mathsf{dv}.\pi, \mathsf{dv}.\mathcal{T})$ *accepts for all pair* $(\mathsf{pk}_P, \mathsf{sk}_P) \in \mathcal{R}_P$ *which is correctly generated and by correction of the PCF and completeness of* Σ*-protocol,* $\Sigma.V(x, a_i, b_i, z_{i,b_i})$ *also accept for all* $i \in [1, \lambda]$. □

Theorem 10 (Knowledge soundness). *If* dv *is adaptive knowledge sound,* Σ *is 2-special sound and* PCF *for OT correlation satisfies strong PK-PCF security property then DV-NIZK scheme from fig. 23 is reusable adaptive knowledge soundness.*

Proof. We describe an efficient simulator Sim *that correctly emulates the verifier without knowing about* sk_V. *The simulator is done as follows:*

- $\mathsf{Sim.Setup}(1^\lambda)$*: compute* $\mathsf{PCF.Setup}(1^\lambda.) \to \mathsf{PCF.pp}$, $\mathsf{PCF.Gen}_1(1^\lambda, \mathsf{PCF.pp}) \to (\mathsf{pk}_V, \mathsf{sk}_V)$ *and* $\mathsf{dv.Setup}(1^\lambda) \to (\mathsf{dv.crs}, \mathsf{dv}.\mathcal{T})$*, define the output* $(\mathsf{PCF.pp}, \mathsf{dv.crs}, \mathsf{pk}_V) \to \mathsf{crs}$*, a trapdoor* $\mathcal{T} := (\mathsf{sk}_V, \mathsf{dv}.\mathcal{T})$ *and erase* sk_V.
- $\mathsf{Sim.V}(\mathsf{crs}, x, \pi, \mathsf{dv}.\mathcal{T})$*: parse* $\pi = (\mathsf{pk}_P, \mathsf{dv}.\pi, \{a_i, m_{i,0}, m_{i,1}\}_{i \le \lambda})$*, use the* $\mathsf{dv.Ext}$ *to extract the* sk_P *i.e* $\mathsf{sk}_P \leftarrow \mathsf{dv.Ext}(\mathsf{crs}, \mathsf{pk}_P, \mathsf{dv}.\mathcal{T})$*. From* sk_P*, for each* $i \in [1, \lambda]$*, compute* $k_0 \leftarrow \mathsf{PCF.KeyDer}(0, \mathsf{sk}_P, \mathsf{pk}_V)$, $(r_{0,i}, r_{1,i}) \leftarrow \mathsf{PCF.Eval}(1^\lambda, 0, k_0, x)$, $(b_i, r_{i,b_i}) \leftarrow \mathsf{RSample}(1^\lambda, 0, (r_{0,i}, r_{1,i}))$ *then define* $(z_{0,i}, z_{1,i}) = (m_{i,0} \oplus r_{0,i}, m_{i,1} \oplus r_{1,i})$.
  Firstly, check that if $(\mathsf{pk}_P, \mathsf{sk}_P) \notin \mathcal{R}_P$ *then outputs 0. Otherwise, continue to check* $\Sigma.V(x, a_i, b_i, z_{i,b_i}) = 1$ *for all* $i \in [1, \lambda]$*. If all checks succeeded, accept (output 1). Otherwise, reject (output 0). To extract a witness, pick* $i \xleftarrow{\$} [1, \lambda]$ *then define* $w \leftarrow \Sigma.\mathsf{Ext}(x, a_i, z_{i,0}, z_{i,1})$.

□

The simulator Sim first calls $\mathsf{Sim.Setup}(1^\lambda)$ to generate crs, and store $\mathsf{dv}.\mathcal{T}$. Each time the $\mathcal{A}$ sends a query (x, π) to the oracle $\mathcal{O}(\mathsf{crs}, ., ., \mathcal{T})$, Sim simulates $\mathcal{O}(\mathsf{crs}, ., ., \mathcal{T})$ (without knowing sk_V) by running $\mathsf{Sim.V}(\mathsf{crs}, x, \pi, \mathsf{dv}.\mathcal{T})$ and outputs whatever Sim.V outputs. When $\mathcal{A}$ outputs a final answer (x^*, π^*), Sim computes a witness w as in Sim.V.

We prove the following claim: for any input (x, π), it hold that

$$\Pr\left[\begin{array}{c} (\mathsf{crs}, \mathcal{T}) \leftarrow \mathsf{Setup}(1^\lambda) \\ b \leftarrow \mathsf{Sim.V}(\mathsf{crs}, x, \pi, \mathsf{dv}.\mathcal{T}) \ : \ b = b' \\ b' \leftarrow \mathsf{V}(\mathsf{crs}, x, \pi, \mathcal{T}) \end{array}\right] \approx 1$$

We show that if $b = 1$, then $b' = 1$ with overwhelming probability. Indeed, if $b = 1$ it means

- $(\mathsf{pk}_P, \mathsf{sk}_P) \in \mathcal{R}_P$ then $\mathsf{dv.V}(\mathsf{dv.crs}, \mathsf{pk}_P, \mathsf{dv}.\pi, \mathsf{dv}.\mathcal{T}) = 1$.
- $\Sigma.V(x, a_i, b_i, z_{i,b_i}) = 1$ for all $i \in [1, \lambda]$ where $b_i \xleftarrow{\$} \{0, 1\}$ (output of Rsample) then by the soundness of Σ-protocol, we have $(x, w) \in \mathcal{R}$ with a probability of at least $1 - 1/2^\lambda$. This leads to $\Sigma.V(x, a_i, b_i, z_{i,b_i}) = 1$ for all $i \in [1, \lambda]$ and $b_i \in \{0, 1\}$ (output of PCF.Eval) with a probability of at least $1 - 1/2^\lambda$.

Therefore, $\mathsf{V}(\mathsf{crs}, x, \pi, \mathcal{T}) = 1$ i.e if Sim's checks succeeding then the verifier's checks necessarily succeed with high probability. In particular, the probability that the Sim accepts the proof while the verifier rejects it is at most $\epsilon_\Sigma = 1/2^\lambda$.

We next prove that if $b' = 1$ then $b = 1$ with high probability. Assume the Sim rejects the proof while the verifier accepts it. Let denote ϵ as the probability of verifier that accepts the proof. Since Sim rejects the proof then at least one of checks must fail: either $(\mathsf{pk}_P, \mathsf{sk}_P) \notin \mathcal{R}_P$ or $\exists i \in [1, \lambda] : \ \Sigma.V(x, a_i, b_i, z_{i,b_i}) = 0$ (b_i is output of RSample). Because the verifier accepts the proof then $\mathsf{dv.V}(\mathsf{dv.crs}, \ \mathsf{pk}_P, \mathsf{dv}.\pi, \mathsf{dv}.\mathcal{T}) = 1$ and $\Sigma.V(x, a_i, b_i, z_{i,b_i}) = 1$ for all $i \in [1, \lambda]$ and b_i is output of PCF.Eval (computed honestly).

- By the knowledge soundness of dv then

$$\Pr\left[(\mathsf{pk}_P, \mathsf{sk}_P) \notin \mathcal{R}_P \middle| \begin{array}{c} \mathsf{dv.V}(\mathsf{dv.crs}, \mathsf{pk}_P, \mathsf{dv}.\pi, \mathsf{dv}.\mathcal{T}) = 1 \\ \mathsf{sk}_P \leftarrow \mathsf{dv.Ext}(\mathsf{crs}, \mathsf{pk}_P, \mathsf{dv}.\pi, \mathsf{dv}.\mathcal{T}) \end{array}\right] = \epsilon_{\mathsf{dv}}$$

- By the security of PCF, $\Sigma.V(x, a_i, b_i, z_{i,b_i}) = 1$ for all $i \in [1, \lambda]$ and b_i is output of $\mathsf{PCF.Eval}$ (computed honestly) can be simulated indistinguishable by $\Sigma.V(x, a_i, b_i, z_{i,b_i}) = 1$ for all $i \in [1, \lambda]$ and b_i is output of $\mathsf{RSample}$ i.e a random bit. After that, since b_i is uniformly random then by the soundness of Σ- protocol, with high probability $(x, w) \in \mathcal{R}$.
  For convenience, we denote the event $\Sigma.V(x, a_i, b_i, z_{i,b_i}) = 1 \forall i \in [1, \lambda]$ where $b_i \xleftarrow{\$} \{0, 1\}$ as Ver and $\exists i \in [1, \lambda] : \ \Sigma.V(x, a_i, b_i, z_{i,b_i}) = 0$ where $b_i \xleftarrow{\$} \{0, 1\}$ as Bad. Then from the security of PCF and the soundness of sigma protocol, we have:

$$\Pr[\mathsf{Ver}] \geq \epsilon - \epsilon_{\mathsf{PCF}} \quad \text{and} \quad \Pr\Big[\ \mathsf{Ver} \Big| \mathsf{Bad}\ \Big] \leq \epsilon_{\Sigma}$$

  Observe that:

$$\begin{aligned}\Pr[\mathsf{Ver}] &= \Pr[\mathsf{Ver} \wedge \mathsf{Bad}] + \Pr[\mathsf{Ver} \wedge \overline{\mathsf{Bad}}] \\ &= \Pr[\mathsf{Ver} \mid \mathsf{Bad}] \cdot \Pr[\mathsf{Bad}] + \Pr[\mathsf{Ver} \wedge \overline{\mathsf{Bad}}] \leq \epsilon_{\Sigma} + \Pr[\mathsf{Ver} \wedge \overline{\mathsf{Bad}}]\end{aligned}$$

  Then we obtain $\Pr[\mathsf{Ver} \wedge \overline{\mathsf{Bad}}] \geq \epsilon - \epsilon_{\mathsf{PCF}} - \epsilon_{\Sigma}$. Putting everything together, the verifier accepts a proof with probability of ϵ then the simulator also accept this proof with probability of at least $\epsilon - \epsilon_{\mathsf{PCF}} - \epsilon_{\Sigma} - \epsilon_{\mathsf{dv}}$.

In conclusion, we have:

$$\Pr\left[\begin{array}{c}(\mathsf{crs}, \mathcal{T}) \leftarrow \mathsf{Setup}(1^{\lambda}) \\ b \leftarrow \mathsf{Sim.V}(\mathsf{crs}, x, \pi, \mathsf{dv}.\mathcal{T}) \ : \ b = b' \\ b' \leftarrow \mathsf{V}(\mathsf{crs}, x, \pi, \mathcal{T})\end{array}\right] \geq 1 - \mu$$

where $\mu = \epsilon_{\mathsf{dv}} + 2.\epsilon_{\Sigma} + \epsilon_{\mathsf{PCF}} = \mathsf{negl}(\lambda)$.

Now consider an $\mathcal{A}$ that outputs an accepting proof with probability of at least ϵ after Q polynomial times queries to oracle $\mathcal{O}(\mathsf{crs}, ., ., \mathcal{T})$. By the above claim, $\mathcal{A}$ outputs an accepting proof with probability of at least $\epsilon - Q.\mu$ after interacting Q times with $\mathsf{Sim.V}(\mathsf{crs}, x, \pi, \mathsf{dv}.\mathcal{T})$; moreover, with probability at least $1 - \mu'$ ($\mu' = \epsilon_{\mathsf{dv}} + \epsilon_{\Sigma} + \epsilon_{\mathsf{PCF}}$), this proof is also accepted by Sim 's verification algorithm. Overall, Sim obtains a proof accepted by his verification algorithm with probability at least $\approx \epsilon - (Q + 1)\mu$. In particular, this implies that w extracted by Sim from π satisfies $(x, w) \in \mathcal{R}$ with probability at least $\epsilon - (Q + 1)\mu$. Therefore, Sim extracts a valid witness with probability at least $\epsilon - (Q + 1)\mu$. As $(Q + 1)\mu = \mathsf{negl}(\lambda)$, we conclude that if $\mathcal{A}$ outputs an accepting proof with non-negligible probability, then Sim extracts a valid witness with non-negligible probability.

Theorem 11 (Zero Knowledge). *If* dv *is adaptive single-theorem ZK,* Σ *is special-honest verifier ZK and strong* PCF *satisfies pseudorandom* $\mathcal{Y}$*-output property then DV-NIZK scheme from fig. 23 is adaptive multi-theorem ZK.*

Proof. Denote Sim_{Σ} *as the efficient simulator for* Σ*-protocol to prove HVZK property. Formally, given a challenger* b_i *ahead of time,* Sim_{Σ} *can output an accepting proof without knowing the witness i.e* $\mathsf{Sim}_{\Sigma}(x, b_i) \to (a, b_i, z_{b_i})$ *such that* $\Sigma.V(x, a, b_i, z_{b_i}) = 1$*. And for (one time) DV-NIZK scheme, there exists a simulator* $\mathsf{Sim}_{\mathsf{dv}} = (S_1^{\mathsf{dv}}, S_2^{\mathsf{dv}})$ *which is used to prove the zero-knowledge of scheme where* $(\mathsf{dv.crs}, \mathsf{dv}.\mathcal{T}) \leftarrow S_1^{\mathsf{dv}}(1^{\lambda})$ *outputs a simulated common reference string and a simulator trapdoor and* $\mathsf{dv}.\pi \leftarrow S_2^{\mathsf{dv}}(\mathsf{dv.crs}, \mathsf{dv}.\mathcal{T}, x)$ *outputs an accepted proof.*

We now construct a zero-knowledge simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ *below:*

- $\mathcal{S}_1(1^{\lambda}) \to (\mathsf{crs}, \mathcal{T})$*. On inputs* 1^{λ} *and outputs* $\mathsf{crs} \leftarrow (\mathsf{PCF.pp}, \mathsf{dv.crs}, \mathsf{pk}_V)$ *and a trapdoor* $\mathcal{T} := (\mathsf{sk}_V, \mathsf{dv}.\mathcal{T})$ *such that*
  - $\mathsf{PCF.Setup}(1^{\lambda}) \to \mathsf{PCF.pp}$, $\mathsf{PCF.Gen}_1(1^{\lambda}, \mathsf{PCF.pp}) \to (\mathsf{pk}_V, \mathsf{sk}_V)$
  - $\mathsf{Sim}_{\mathsf{dv}}^1(1^{\lambda}) \to (\mathsf{dv.crs}, \mathsf{dv}.\mathcal{T})$
- $\mathcal{S}_2(\mathsf{crs}, \mathcal{T}, x) \to \pi$*. On inputs* $(\mathsf{crs}, \mathcal{T}, x)$*,* $\mathcal{S}_2$ *randomly chooses a pair* $(\mathsf{pk}_P, \mathsf{sk}_P) \in \mathcal{R}_P$*, computes* $\mathsf{PCF.KeyDer}(1, \mathsf{sk}_V, \mathsf{pk}_P) \to k_1$ *and for each invocation* $i \in [1, \lambda]$*, computes as below:*

$$\mathsf{PCF.Eval}(1^{\lambda}, 1, k_1, x) = (b_i, r_{i,b_i})$$

$\mathcal{S}_2$ uses the simulator $\mathsf{Sim}_\Sigma(x, b_i)$ to get an accepting transcript (a_i, b_i, z_{i,b_i}) for each $i \in [1, \lambda]$. $\mathcal{S}_2$ now defines the output:

$$\pi := (\mathsf{pk}_P, \mathsf{dv}.\pi, \{a_i, m_{i,0}, m_{i,1}\}_{i \leq \lambda})$$

where $\mathsf{dv}.\pi = \mathsf{dv}.\mathsf{P}(\mathsf{dv.crs}, \mathsf{pk}_P, \mathsf{sk}_P)$ and for each $i \in [1, \lambda]$, $m_{i,b_i} = z_{i,b_i} \oplus r_{i,b_i}$ otherwise $m_{i,1-b_i}$ is picked randomly.

To complete the proof, we consider a sequence of hybrid experiments:

- Hyb_0. *Namely, at the beginning of the game, the challenger samples $\Pi.\mathsf{Setup}(1^\lambda) \to (\mathsf{crs}, \mathcal{T})$ and gives crs to the adversary, where $\mathsf{crs} \leftarrow (\mathsf{PCF.pp}, \mathsf{dv.crs}, \mathsf{pk}_V)$ and $\mathcal{T} := (\mathsf{sk}_V, \mathsf{dv}.\mathcal{T})$. When the adversary makes a verification query on $(x, w) \in \mathcal{R}$, the challenger replies with $\Pi.\mathsf{P}(\mathsf{crs}, x, w) = (\mathsf{pk}_P, \mathsf{dv}.\pi, \{a_i, m_{i,0}, m_{i,1}\}_{i \leq \lambda})$.*
- Hyb_1. *Same as Hyb_0, except we replace:*
  - *$(\mathsf{dv.crs}, \mathsf{dv}.\mathcal{T}) \in \Pi.\mathsf{Setup}(1^\lambda)$ by $(\mathsf{dv.crs}', \mathsf{dv}.\mathcal{T}') \leftarrow \mathcal{S}_1^{\mathsf{dv}}(1^\lambda)$*
  - *$\mathsf{dv}.\pi \in \pi$ by $\mathsf{dv}.\pi' \leftarrow S_2^{\mathsf{dv}}(\mathsf{dv.crs}', \mathsf{dv}.\mathcal{T}', \mathsf{pk}_P)$*

  This is computational indistinguishable from Hyb_0 by the the zero-knowledge of dv.
- Hyb_2. *Same as Hyb_1, except the challenger simulates $\{a'_i, (m'_{i,0}, m'_{i,1})_{i \leq \lambda}\} \in \pi$ by the following way. For each $i \in [1, \lambda]$,*
  - *Compute $k_1 \leftarrow \mathsf{PCF.KeyDer}(1, \mathsf{sk}_V, \mathsf{pk}_P)$, $(b_i, r_{i,b_i}) \leftarrow \mathsf{PCF.Eval}(1^\lambda, 1, k_1, x)$*
  - *Use Sim_Σ for simulating (a'_i, c'_{i,b_i}) i.e compute $(a'_i, b_i, c'_{i,b_i}) \leftarrow \mathsf{Sim}_\Sigma(x, b_i)$*
  - *Define $m'_{i,b_i} := c'_{i,b_i} \oplus r_{i,b_i}$ and $m'_{i,1-b_i}$ is picked randomly*

  This is computationally indistinguishable from Hyb_1 by the pseudorandom output of PCF and HVZK of Σ-protocol. Indeed, the adversary can only compute the value of (b_i, r_{i,b_i}) by the $\mathsf{PCF.Eval}$ and from the view of adversary $r_{i,1-b_i}$ is pseudorandom then $m_{i,1-b_i}$ also for all $i \in [1, \lambda]$. Moreover, for all $(x, w) \in \mathcal{R}$, with the $\Pi.\mathsf{Setup}(1^\lambda) := (\mathsf{crs}', \mathcal{T}')$ where $\mathsf{crs}' = \mathsf{PCF.pp}, \mathsf{dv.crs}', \mathsf{pk}_V)$, $\mathcal{T}' = (\mathsf{sk}_V, \mathsf{dv}.\mathcal{T}')$ and prove $\pi' = (\mathsf{pk}_P, \mathsf{dv}.\pi', \{a'_i, m'_{i,0}, m'_{i,1}\}_{i \leq \lambda})$, the adversary always accepts it i.e $\Pi.\mathsf{V}(\mathsf{crs}', x, \mathcal{T}', \pi') = 1$.

□

7.2 Concrete Instantiation

Following our PK-PCF construction (fig. 19), we define the language $\mathcal{L}_p$ for non-reusable DV-NIZK argument as below

$$\mathcal{L}_p := \left(G^t, H^t(1+N)^{r'}; \left(h^{r^s}, (h')^{r^s}\right)_{s \in S} \right)$$

where $\mathsf{sk}_p := (h, h', r, t)$ and $r := g_q^{r'} \mod p$.
To build a non-reusable DV-NIZK argument of knowledge, firstly we build a Σ protocol for language $\mathcal{L}_p$ then later using the construction of [PsV06] with the trapdoor $\mathsf{dv}.\mathcal{T}$ as the list of sk which are used in public-key encryption to obtain a (non-reusable) DV-NIZK scheme that satisfies adaptive knowledge soundness, zero-knowledge.
For instantiation of *strong* PK-PCF, we use our PK-PCF construction (fig. 19) plus the techinique [BCE+23] that can convert weak PCF to strong PCF using PRF. We highlight our PK-PCF satisfies the strong security about Supp which we showed directly from the proof security in theorem 7.

Acknowledgments

Geoffroy Couteau and Dung Bui were supported by the French Agence Nationale de la Recherche (ANR), under grant ANR-20-CE39-0001 (project SCENE), and by the France 2030 ANR Project ANR22-PECY-003 SecureCompute. Dung Bui was supported by DIM Math Innovation 2021 (N°IRIS: 21003816) from the Paris Mathematical Sciences Foundation (FSMP) funded by the Paris Ile-deFrance Region. Pierre Meyer was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grants agreement number 852952 (HSS) and 803096 (SPEC). Alain Passelègue and Mahshid Riahinia were supported by the French ANR RAGE project (ANR-20-CE48-0011) and the France 2030 ANR Project (ANR22-PECY-003) SecureCompute.

References

ABP15. Michel Abdalla, Fabrice Benhamouda, and Alain Passelègue. An algebraic framework for pseudorandom functions and applications to related-key security. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 388–409. Springer, Heidelberg, August 2015.

ABR16. Benny Applebaum, Andrej Bogdanov, and Alon Rosen. A dichotomy for local small-bias generators. *Journal of Cryptology*, 29(3):577–596, July 2016.

ADDG23. Martin R. Albrecht, Alex Davidson, Amit Deo, and Daniel Gardham. Crypto dark matter on the torus: Oblivious PRFs from shallow PRFs and FHE. Cryptology ePrint Archive, Report 2023/232, 2023. https://eprint.iacr.org/2023/232.

AHI11. Benny Applebaum, Danny Harnik, and Yuval Ishai. Semantic security under related-key attacks and applications. In Bernard Chazelle, editor, *Innovations in Computer Science - ICS 2011, Tsinghua University, Beijing, China, January 7-9, 2011. Proceedings*, pages 45–60. Tsinghua University Press, 2011.

AK19. Benny Applebaum and Eliran Kachlon. Sampling graphs without forbidden subgraphs and unbalanced expanders with negligible error. In David Zuckerman, editor, *60th FOCS*, pages 171–179. IEEE Computer Society Press, November 2019.

AKPW13. Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited - new reduction, properties and applications. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 57–74. Springer, Heidelberg, August 2013.

AL16. Benny Applebaum and Shachar Lovett. Algebraic attacks against random local functions and their countermeasures. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 1087–1100. ACM Press, June 2016.

AMN+18. Nuttapong Attrapadung, Takahiro Matsuda, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Constrained PRFs for NC^1 in traditional groups. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 543–574. Springer, Heidelberg, August 2018.

App12. Benny Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 805–816. ACM Press, May 2012.

App15. Benny Applebaum. The cryptographic hardness of random local functions – survey. Cryptology ePrint Archive, Report 2015/165, 2015. https://eprint.iacr.org/2015/165.

App17. Benny Applebaum. Exponentially-hard gap-CSP and local PRG via local hardcore functions. In Chris Umans, editor, *58th FOCS*, pages 836–847. IEEE Computer Society Press, October 2017.

AR16. Benny Applebaum and Pavel Raykov. Fast pseudorandom functions based on expander graphs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part I*, volume 9985 of *LNCS*, pages 27–56. Springer, Heidelberg, October / November 2016.

BCE+23. Chris Brzuska, Geoffroy Couteau, Christoph Egger, Pihla Karanko, and Pierre Meyer. New random oracle instantiations from extremely lossy functions. Cryptology ePrint Archive, Paper 2023/1145, 2023. https://eprint.iacr.org/2023/1145.

BCG+17. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, and Michele Orrù. Homomorphic secret sharing: Optimizations and applications. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2105–2122. ACM Press, October / November 2017.

BCG+19a. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 291–308. ACM Press, November 2019.

BCG+19b. 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, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 489–518. Springer, Heidelberg, August 2019.

BCG+20. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Correlated pseudorandom functions from variable-density LPN. In *61st FOCS*, pages 1069–1080. IEEE Computer Society Press, November 2020.

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, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 603–633. Springer, Heidelberg, August 2022.

BCGI18. Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 896–912. ACM Press, October 2018.

BCP03. Emmanuel Bresson, Dario Catalano, and David Pointcheval. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In Chi-Sung Laih, editor, *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 37–54. Springer, Heidelberg, November / December 2003.

Bea92. Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 420–432. Springer, Heidelberg, August 1992.

Bea95. Donald Beaver. Precomputing oblivious transfer. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 97–109. Springer, Heidelberg, August 1995.

Ber06. Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 207–228. Springer, Heidelberg, April 2006.

BFKL94. Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 278–291. Springer, Heidelberg, August 1994.

BGMM20. James Bartusek, Sanjam Garg, Daniel Masny, and Pratyay Mukherjee. Reusable two-round MPC from DDH. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 320–348. Springer, Heidelberg, November 2020.

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, Part II*, volume 11240 of *LNCS*, pages 699–729. Springer, Heidelberg, November 2018.

Blu86. Manuel Blum. How to prove a theorem so no one else can claim it, August 1986. Invited 45 minute address to the International Congress of Mathematicians, 1986. To appear in the Proceedings of ICM 86.

BM90. Mihir Bellare and Silvio Micali. Non-interactive oblivious transfer and applications. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 547–557. Springer, Heidelberg, August 1990.

BPR12. Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Heidelberg, April 2012.

BQ09. Andrej Bogdanov and Youming Qiao. On the security of goldreich's one-way function. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 392–405. Springer, 2009.

BV15. Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic PRFs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 1–30. Springer, Heidelberg, March 2015.

CC18. Pyrros Chaidos and Geoffroy Couteau. Efficient designated-verifier non-interactive zero-knowledge proofs of knowledge. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 193–221. Springer, Heidelberg, April / May 2018.

CCKK21. Jung Hee Cheon, Wonhee Cho, Jeong Han Kim, and Jiseung Kim. Adventures in crypto dark matter: Attacks and fixes for weak pseudorandom functions. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 739–760. Springer, Heidelberg, May 2021.

CDM+18. Geoffroy Couteau, Aurélien Dupin, Pierrick Méaux, Mélissa Rossi, and Yann Rotella. On the concrete security of Goldreich's pseudorandom generator. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 96–124. Springer, Heidelberg, December 2018.

CDS94. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, Heidelberg, August 1994.

CEMT14. James Cook, Omid Etesami, Rachel Miller, and Luca Trevisan. On the one-way function candidate proposed by goldreich. *ACM Transactions on Computation Theory (TOCT)*, 6(3):14, 2014.

CH19. Geoffroy Couteau and Dennis Hofheinz. Designated-verifier pseudorandom generators, and their applications. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 562–592. Springer, Heidelberg, May 2019.

CH20. Geoffroy Couteau and Dominik Hartmann. Shorter non-interactive zero-knowledge arguments and ZAPs for algebraic languages. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 768–798. Springer, Heidelberg, August 2020.

CHH+07. Ronald Cramer, Goichiro Hanaoka, Dennis Hofheinz, Hideki Imai, Eike Kiltz, Rafael Pass, abhi shelat, and Vinod Vaikuntanathan. Bounded CCA2-secure encryption. In Kaoru Kurosawa,

editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 502–518. Springer, Heidelberg, December 2007.

CJJQ23. Geoffroy Couteau, Abhishek Jain, Zhengzhong Jin, and Willy Quach. A note on non-interactive zero-knowledge from cdh. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 731–764, Cham, 2023. Springer Nature Switzerland.

CKLR21. Geoffroy Couteau, Michael Klooß, Huang Lin, and Michael Reichle. Efficient range proofs with transparent setup from bounded integer commitments. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part III*, volume 12698 of *LNCS*, pages 247–277. Springer, Heidelberg, October 2021.

CL15. Craig Costello and Patrick Longa. Four$\mathbb{Q}$: Four-dimensional decompositions on a $\mathbb{Q}$-curve over the Mersenne prime. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 214–235. Springer, Heidelberg, November / December 2015.

CM01. Mary Cryan and Peter Bro Miltersen. On pseudorandom generators in nc 0. In *International Symposium on Mathematical Foundations of Computer Science*, pages 272–284. Springer, 2001.

CMPR23. Geoffroy Couteau, Pierre Meyer, Alain Passelègue, and Mahshid Riahinia. Constrained pseudorandom functions from homomorphic secret sharing. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 194–224. Springer, Heidelberg, April 2023.

CNs07. Jan Camenisch, Gregory Neven, and abhi shelat. Simulatable adaptive oblivious transfer. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 573–590. Springer, Heidelberg, May 2007.

Cré88. Claude Crépeau. Equivalence between two flavours of oblivious transfers. In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 350–354. Springer, Heidelberg, August 1988.

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, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 502–534, Virtual Event, August 2021. Springer, Heidelberg.

CS02. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Heidelberg, April / May 2002.

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, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 517–547, Virtual Event, August 2021. Springer, Heidelberg.

DGS03. Ivan Damgård, Jens Groth, and Gorm Salomonsen. *The Theory and Implementation of an Electronic Voting System*, pages 77–99. Springer US, Boston, MA, 2003.

DILO22. Samuel Dittmer, Yuval Ishai, Steve Lu, and Rafail Ostrovsky. Authenticated garbling from simple correlations. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 57–87. Springer, Heidelberg, August 2022.

DMMS21. Sébastien Duval, Pierrick Méaux, Charles Momin, and François-Xavier Standaert. Exploring crypto-physical dark matter and learning with physical rounding. *IACR TCHES*, 2021(1):373–401, 2021. https://tches.iacr.org/index.php/TCHES/article/view/8738.

DMR23. Aurélien Dupin, Pierrick Méaux, and Mélissa Rossi. On the algebraic immunity—resiliency trade-off, implications for goldreich's pseudorandom generator. *Designs, Codes and Cryptography*, pages 1–45, 2023.

DNNR17. Ivan Damgård, Jesper Buus Nielsen, Michael Nielsen, and Samuel Ranellucci. The TinyTable protocol for 2-party secure computation, or: Gate-scrambling revisited. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 167–187. Springer, Heidelberg, August 2017.

DPSZ12. Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 643–662. Springer, Heidelberg, August 2012.

FKOS15. Tore Kasper Frederiksen, Marcel Keller, Emmanuela Orsini, and Peter Scholl. A unified approach to MPC with preprocessing using OT. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 711–735. Springer, Heidelberg, November / December 2015.

GGM84a. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th FOCS*, pages 464–479. IEEE Computer Society Press, October 1984.

GGM84b. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 276–288. Springer, Heidelberg, August 1984.

GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

Gol00. Oded Goldreich. Candidate one-way functions based on expander graphs. Cryptology ePrint Archive, Report 2000/063, 2000. https://eprint.iacr.org/2000/063.

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.

HSS20. Carmit Hazay, Peter Scholl, and Eduardo Soria-Vazquez. Low cost constant round MPC combining BMR and oblivious transfer. *Journal of Cryptology*, 33(4):1732–1786, October 2020.

IKNP03. Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161. Springer, Heidelberg, August 2003.

JJ21. Abhishek Jain and Zhengzhong Jin. Non-interactive zero knowledge from sub-exponential DDH. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 3–32. Springer, Heidelberg, October 2021.

JLS21. Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *53rd ACM STOC*, pages 60–73. ACM Press, June 2021.

JMN23. Thomas Johansson, Willi Meier, and Vu Nguyen. Differential cryptanalysis of mod-2/mod-3 constructions of binary weak prfs. In *2023 IEEE International Symposium on Information Theory (ISIT)*, pages 477–482. IEEE, 2023.

KKRT16. Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 818–829. ACM Press, October 2016.

KNYY19. Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Designated verifier/prover and preprocessing NIZKs from Diffie-Hellman assumptions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 622–651. Springer, Heidelberg, May 2019.

KW15. Eike Kiltz and Hoeteck Wee. Quasi-adaptive NIZK for linear subspaces revisited. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 101–128. Springer, Heidelberg, April 2015.

LPSY15. Yehuda Lindell, Benny Pinkas, Nigel P. Smart, and Avishay Yanai. Efficient constant round multi-party computation combining BMR and SPDZ. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 319–338. Springer, Heidelberg, August 2015.

LV17. Alex Lombardi and Vinod Vaikuntanathan. Limits on the locality of pseudorandom generators and applications to indistinguishability obfuscation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 119–137. Springer, Heidelberg, November 2017.

Méa. P Méaux. On the fast algebraic immunity of threshold functions. crypt. commun. 13 (5), 741–762 (2021).

Méa22. Pierrick Méaux. On the algebraic immunity of direct sum constructions. *Discrete Applied Mathematics*, 320:223–234, 2022.

MST03. Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On e-biased generators in NC0. In *44th FOCS*, pages 136–145. IEEE Computer Society Press, October 2003.

Nec94. V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994.

NNOB12. Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 681–700. Springer, Heidelberg, August 2012.

NR95. Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. In *36th FOCS*, pages 170–181. IEEE Computer Society Press, October 1995.

NR97. Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th FOCS*, pages 458–467. IEEE Computer Society Press, October 1997.

OST19. Igor Carboni Oliveira, Rahul Santhanam, and Roei Tell. Expander-based cryptography meets natural proofs. In Avrim Blum, editor, *ITCS 2019*, volume 124, pages 18:1–18:14. LIPIcs, January 2019.

OSY21. Claudio Orlandi, Peter Scholl, and Sophia Yakoubov. The rise of paillier: Homomorphic secret sharing and public-key silent OT. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 678–708. Springer, Heidelberg, October 2021.

OW14. Ryan ODonnell and David Witmer. Goldreich's prg: evidence for near-optimal polynomial stretch. In *Computational Complexity (CCC), 2014 IEEE 29th Conference on*, pages 1–12. IEEE, 2014.

Pai99. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Heidelberg, May 1999.

Ped92. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.

PsV06. Rafael Pass, abhi shelat, and Vinod Vaikuntanathan. Construction of a non-malleable encryption scheme from any semantically secure one. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 271–289. Springer, Heidelberg, August 2006.

QRW19. Willy Quach, Ron D. Rothblum, and Daniel Wichs. Reusable designated-verifier NIZKs for all NP from CDH. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 593–621. Springer, Heidelberg, May 2019.

RRT23. Srinivasan Raghuraman, Peter Rindal, and Titouan Tanguy. Expand-convolute codes for pseudorandom correlation generators from lpn. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 602–632, Cham, 2023. Springer Nature Switzerland.

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

Üna23a. Akin Ünal. New baselines for local pseudorandom number generators by field extensions. *Cryptology ePrint Archive*, 2023.

Üna23b. Akin Ünal. Worst-case subexponential attacks on PRGs of constant degree or constant locality. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part I*, volume 14004 of *LNCS*, pages 25–54. Springer, Heidelberg, April 2023.

WRK17. Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Global-scale secure multiparty computation. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 39–56. ACM Press, October / November 2017.

YGJL21. Jing Yang, Qian Guo, Thomas Johansson, and Michael Lentmaier. Revisiting the concrete security of goldreich's pseudorandom generator. *IEEE Transactions on Information Theory*, 68(2):1329–1354, 2021.