Robust Additive Randomized Encodings from IO and Pseudo-Non-linear Codes*

Nir Bitansky[†] Sapir Freizeit[‡]

Tel Aviv University

February 2024

Abstract

Additive randomized encodings (ARE), introduced by Halevi, Ishai, Kushilevitz, and Rabin (CRYPTO 2023), reduce the computation of a k-party function $f(x_1, \ldots, x_k)$ to locally computing encodings $\hat{x}_i$ of each input x_i and then adding them together over some Abelian group into an output encoding $\hat{y} = \sum \hat{x}_i$, which reveals nothing but the result. In *robust* ARE (RARE) the sum of any subset of $\hat{x}_i$, reveals only the residual function obtained by restricting the corresponding inputs. The appeal of (R)ARE comes from the simplicity of the online part of the computation, involving only addition, which yields for instance non-interactive multi-party computation in the *shuffle model* where messages from different parties are anonymously shuffled. Halevi, Ishai, Kushilevitz, and Rabin constructed ARE from standard assumptions and RARE in the ideal obfuscation model, leaving open the question of whether RARE can be constructed in the plain model.

We construct RARE in the plain model from indistinguishability obfuscation, which is necessary, and a new primitive that we call *pseudo-non-linear codes*. We provide two constructions of this primitive assuming either Learning with Errors or Decision Diffie Hellman. A bonus feature of our construction is that it is *online succinct*. Specifically, encodings $\hat{x}_i$ can be decomposed to offline parts $\hat{z}_i$ that can be sent directly to the evaluator and short online parts $\hat{g}_i$ that are added together.

*Supported in part by the European Research Council (ERC) under the European Union's Horizon Europe research and innovation programme (grant agreement No. 101042417, acronym SPP).

[†]nbitansky@gmail.com.

[‡]sapirfreizeit@gmail.com

Contents

1 Introduction

The notion of *randomized encodings* [AIK06] has been extremely influential in cryptography. In the context of secure multi-party computation (MPC) [Yao86, GMW87, BOGW88, CCD88], different notions of randomized encodings have been introduced to push MPC to the limit in terms of efficiency and communication complexity. Recently, Halevi, Ishai, Kushilevitz, and Rabin [HIKR23] introduced the notion of *additive randomized encodings* (ARE). Such randomized encodings can be seen as a version of multi-party randomized encodings [ABT21] where securely computing a multi-party function is reduced to locally computing a randomized encodings of each input and then summing up the encodings over some group $\mathbb{G}$.

Specifically, for a k-party function $f(\boldsymbol{x})$, where $\boldsymbol{x} = (x_1, \ldots, x_k)$, input encodings $\hat{x}_i \leftarrow \mathsf{Enc}_{pp}(x_i, i)$ are computed using some public parameters pp and reside in an Abelian additive group $\mathbb{G}$. The input encodings are combined to an encoding of the output by addition $\hat{f}(\boldsymbol{x}) = \sum_{i\in[k]} \hat{x}_i$. The output encoding $\hat{f}(\boldsymbol{x})$ can be used to decode $f(\boldsymbol{x})$ but leaks no other information about $\boldsymbol{x}$. Accordingly, the local encoding of each input can be done by each party *offline*, whereas the *online* part of the computation only involves the simple addition function.

As noted in [HIKR23], the addition operation can be realized for instance in the *shuffle model* [IKOS06], where parties can simultaneously send (multiple) anonymous messages to a receiver. This makes the notion of ARE especially appealing allowing for non-interactive MPC (NIMPC) [BGI+14] in a model that does not require correlated randomness or public-key infrastructure.

Robust ARE. The vanilla notion of ARE only guarantees security when the evaluator of $\hat{f}(\boldsymbol{x})$ does not collude with any of the parties. In this work we focus on a stronger notion defined in [HIKR23] of *robust additive randomized encodings* (RARE) that also offers security against insiders. Here a corrupted coalition of parties C obtains the partial sum $\hat{s}_H = \sum_{i\in H} \hat{x}_i$ of encodings of honest parties $H = [k] \setminus C$. This inevitably allows the corrupted coalition to evaluate the residual function $f_{H,\boldsymbol{x}_H}(\boldsymbol{y}_C) = f(\boldsymbol{x}_H, \boldsymbol{y}_C)$ determined by the honest inputs $\boldsymbol{x}_H$ (by encoding inputs $\boldsymbol{y}_C$ of their choice and decoding $\hat{s}_H + \sum_{i\in C} \hat{y}_i$).

Accordingly, and similarly to other notions of robust NIMPC [BGI+14, HIJ+17], the best possible guarantee is hiding everything but this residual function. Perhaps the most intuitive requirement here would be that the partial sum encoding $\hat{s}_H$ could be efficiently simulated given oracle access to $f_{H,\boldsymbol{x}_H}$. However, as observed in [HIKR23], this notion of robustness in fact implies ideal obfuscation and is generally impossible [BGI+01].[1] Hence, and analogously to the regime of obfuscation, we require indistinguishability rather than simulation. That is, the partial sum encoding $\hat{s}_H$ corresponding to honest inputs $\boldsymbol{x}_H$ is computationally indistinguishable from $\hat{s}'_H$ corresponding to honest inputs $\boldsymbol{x}'_H$ provided that the residual functions are the same $f_{H,\boldsymbol{x}_H} \equiv f_{H,\boldsymbol{x}'_H}$. Indeed, this relaxed notion is not subject to any known barriers, and only implies indistinguishability obfuscation, which by now can be constructed from standard assumptions [JLS21].

As part of their comprehensive study of ARE, the authors of [HIKR23] provide a construction of vanilla ARE based on a DDH-type assumption in bilinear groups as well as a construction of simulation-based RARE in the ideal obfuscation model (building also on resettable MPC [GS09, GM11]). Constructing indistinguishability-based RARE in the plain model was left as an open question.

Our Contribution. We construct (indistinguishability-based) RARE in the plain model under the necessary assumption of indistinguishability obfuscation, and a new primitive that we call *pseudo-non-linear codes* (PNLC). We construct such codes based on either Decision Diffie Hellman (DDH) or Learning with Errors (LWE) with a quasi-polynomial noise-to-modulus ratio.

Theorem 1.1 (Informal). *Assuming indistinguishability obfuscation and either DDH or LWE, there exists (indistinguishability based) RARE.*

Online Succinctness. As an added bonus our construction of RARE is *online succinct*. Specifically, our input encodings $\hat{x}_i$ can be decomposed into two parts: an offline part $\hat{z}_i \in \{0,1\}^*$ and an online part $\hat{g}_i \in \mathbb{G}$ whose

[1] To see how this implies obfuscation, consider a two-party RARE where the encoding of one party $\hat{P}$ is though of as an obfuscated program, and the encoding of the second party $\hat{x}$ corresponds to an input x to the program.

size depends only on the security parameter (but not on the input size or the function complexity). The output encoding $\hat{f}(\boldsymbol{x})$ consists of $\hat{z}_1, \dots, \hat{z}_k, \hat{g} = \sum \hat{g}_i$. In particular only the short online encodings $\hat{g}_i$ need to be securely summed together, whereas the bulk of the encoding $\hat{z}_1, \dots, \hat{z}_k$ is done completely offline and can be sent directly to the evaluator.

1.1 Technical Overview

We now give an overview of our techniques, and in particular further discuss the notion of pseudo-non-linear codes.

Warmup: Construction from Ideal Obfuscation. As a warmup, let us first describe how to obtain a RARE based on ideal obfuscation. As mentioned above, such a construction is described in [HIKR23], relying also on resettable MPC. However, we describe here a different construction relying on CCA2 encryption, which will eventually lead to our construction in the plain model.

In this construction, the public parameters pp consist of k public keys $pk_1 \dots pk_k$ for the CCA2 scheme as well as an obfuscated decoding program $\tilde{\mathsf{D}}$, which we will describe in a bit. To encode the input x_i, party i samples a random group element $g_i \leftarrow \mathbb{G}$. It generates as the offline part $\hat{z}_i$ an encryption $ct_i \leftarrow \mathsf{Enc}_{pk_i}(x_i, g_i)$ of the input along with the group element and the online part is simply g_i The encoding $\hat{f}(\boldsymbol{x})$ corresponding to the input $\boldsymbol{x} = (x_1, \dots, x_k)$ consists of

$$\hat{z}_1, \dots, \hat{z}_k, g = \sum_{i \in [k]} g_i \ .$$

The decoding program D has the corresponding secret keys $sk_1, \dots, sk_k$ hardwired, and given $(ct_1, \dots, ct_k, g)$ it decrypts the ciphertexts, obtains the corresponding inputs x_i and group elements g_i, checks that g matches $\sum g_i$, and if so outputs $f(\boldsymbol{x})$.

An adversary that corrupts a subset $C = [k] \setminus H$ obtains the corresponding ciphertexts $\boldsymbol{ct}^*_H = (ct^*_i : i \in H)$ encrypting $(\boldsymbol{x}^*_H, \boldsymbol{g}^*_H)$, the partial sum $g^*_H = \sum_{i \in H} g^*_i$, and black-box access to the decoding program D. To diverge from the residual function $f(\boldsymbol{x}^*_H, \cdot)$, the adversary aims to replace some subset of the honest inputs $\boldsymbol{x}^*_H$. Intuitively, the only attack that the adversary can mount is to choose a subset $\emptyset \subsetneq F \subsetneq H$, fix the honest ciphertexts $\boldsymbol{ct}^*_F$, and complete them with ciphertexts $\boldsymbol{ct}_{\overline{F}}$ encrypting some $(\boldsymbol{x}_{\overline{F}}, \boldsymbol{g}_{\overline{F}})$ of its choice, as well as a group element $g = \sum_{i \in F} g^*_i + \sum_{i \in \overline{F}} g_i$ that matches the sum, and feed $(\boldsymbol{ct}^*_F, \boldsymbol{ct}_{\overline{F}}, g)$ to the decoding oracle. However, an adversary that manages to perform the latter can be used to mount an attack on the CCA2 security of the encryption, as the reduction can decrypt $\boldsymbol{ct}_{\overline{F}}$ and use g to learn $\sum_{i \in F} g^*_i$.

Replacing Ideal Obfuscation with IO. Recall that in contrast to ideal obfuscation where $\tilde{\mathsf{D}}$ behaves as a black box oracle, in the case of IO all that we are guaranteed is that obfuscations of two (equal-size) programs are computationally indistinguishable only as long as there do not exist inputs on which the programs disagree. Indeed, replacing the ideally obfuscated $\tilde{\mathsf{D}}$ with an IO-obfuscated $\tilde{\mathsf{D}}$ encounters two main difficulties.

The first difficulty is that general CCA2 encryption works well with ideal oracles, but is not necessarily IO-friendly. This difficulty, however, is quite standard by now and has been handled before, for instance in IO-based constructions of functional encryption [GGH+13]. Specifically, using the Naor-Yung *double encryption paradigm* [NY90], along with a statistically simulation-sound non-interactive zero knowledge [Sah99], does the trick.

The second difficulty is more unique to the problem at hand and requires new ideas (in fact, a new primitive). Going through the motions of IO and the Naor-Yung paradigm, at some point in our analysis we reach a hybrid where the inputs $\boldsymbol{x}^*_H$ and corresponding group elements $\boldsymbol{g}^*_H$ are hardwired into our obfuscated program and are only ever used when some challenge ciphertexts $\boldsymbol{ct}^*_F$, for $F \subseteq H$, are combined with ciphertexts $\boldsymbol{ct}_{\overline{F}}$ and a consistent group element $g = \sum_{i \in F} g^*_i + \sum_{i \in \overline{F}} g_i$ (mirroring the previously described attack pattern). The problem is that as far as we know, the group elements $\boldsymbol{g}^*_H$ may leak. Here a natural step toward a solution is to use some form of homomorphic one way encoding $\hat{\boldsymbol{g}}^*_H$ (e.g. based on Discrete Logs) that will guarantee that $\boldsymbol{g}^*_H$ remain hidden and yet support the sum check functionality. This alone, however, is not enough, since even if the consistent g is *hard to find* it nevertheless *exists*. Thus, replacing $\boldsymbol{x}^*_H$ with an equivalent $\boldsymbol{x}_H$ (with respect to the residual function) will still change the functionality of the program and will not be protected by IO. This is where the new primitive of pseudo-non-linear codes enters.

Pseudo-non-linear Codes. Roughly speaking, a PNLC scheme allows to encode group elements $g_1, \ldots, g_k$ so that their (possibly randomized) encodings $\hat{g}_1, \ldots, \hat{g}_k$ can be homomorphically added together to yield an encoding $\hat{g}$ of their sum $g = \sum_{i \in [k]} g_i$. At the same time, for any $H \subseteq [k]$, a simulator can generate $|H|$ fake encodings $\tilde{\boldsymbol{\sigma}}_H$ along with a group element τ, such that:

- On one hand, $(\tilde{\boldsymbol{\sigma}}_H, \tau)$ are computationally indistinguishable from $(\hat{\boldsymbol{g}}_H, g)$, where $\hat{\boldsymbol{g}}_H$ are real encodings of *random* group elements $\boldsymbol{g}_H$ along with their sum $g = \sum_{i \in H} g_i$.
- On the other hand, $\tilde{\boldsymbol{\sigma}}_H$ are not truly linear. For any (non-empty) strict subset of fake encodings, their (homomorphic) sum evades the code. (Note that the homomorphic sum of *all* fake encodings $\sum_{i \in H} \tilde{\sigma}_i$ does yield an encoding $\hat{\tau}$ of τ due to indistinguishability and linearity of the real code.)

Equipped with these encodings we are able to finish the proof. We indistinguishably move to fake encodings, and can now claim that unless the attacker uses all ciphertexts $\boldsymbol{ct}^*_H$, the sum-consistent group element g is not only hard to find, but in fact does not exist.

Constructing PNLC. We give two simple constructions of PNLC from LWE and Decision Diffie Hellman. We sketch here the one from LWE. In this construction the public parameters consist of an LWE matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and are associated with the group $\mathbb{G} = \mathbb{Z}_q^n$. An encoding of a group element $\mathbf{s} \in \mathbb{Z}_q^n$ is essentially an LWE encoding $\mathbf{As} + \mathbf{e}$ where $\mathbf{e}$ is taken form an appropriate B-bounded noise distribution. We treat encodings as equivalent if they are close to each other in ℓ_∞ norm. Here encodings are naturally homomorphic by the linearity of the code $\mathbf{A}$, and as long as we add at most k encodings and make sure that $kB \ll q$.

The simulator is also simple, given H of size h, it outputs

$$\tilde{\boldsymbol{\sigma}}_H = \left(\mathbf{u}_1 + \mathbf{e}_1, \ldots, \mathbf{u}_{h-1} + \mathbf{e}_{h-1}, \mathbf{As} - \sum_{i \in [h-1]} \mathbf{u}_i + \mathbf{e}_h \right), \qquad \tau = \mathbf{s} \ ,$$

where $\mathbf{s}$ is a random vector in $\mathbb{Z}_q^n$, $\mathbf{u}_1, \ldots, \mathbf{u}_{h-1}$ are all random vectors in $\mathbb{Z}_q^m$, and each $\mathbf{e}_i$ is B-bounded noise.

Using homomorphism and noise smudging (or flooding), it is not difficult to show a reduction to LWE with a quasi-polynomial noise-to-modulus ratio. As for non-linearity of fake encodings, note that any strict subset of vectors of $\tilde{\boldsymbol{\sigma}}_H$ is completely random and hence their sum is far from the code given by $\mathbf{A}$ with overwhelming probability (provided that $m \approx n \log q$).

We refer the reader to Section 3 for more details and for the DDH-based construction.

2 Preliminaries

- We denote vectors by boldface symbols $\boldsymbol{x}$, and their coordinates by lower case symbols x_i. For a vector $\boldsymbol{x} \in X^k$, and a subset of indices $H \subset [k]$, we denote by $\boldsymbol{x}_H$ its restriction to the coordinates in H. For $\overline{H} = [k] \setminus H$, we denote by $(\boldsymbol{x}_H, \boldsymbol{y}_{\overline{H}})$ the k-coordinate vector $\boldsymbol{z}$ such that $z_i = x_i$ for $i \in H$, and $z_i = y_i$ for $i \in \overline{H}$.
- For a distribution D, $x \leftarrow D$ denotes x from D. For an integer m we denote the distribution of sampling a vector consists of m independent copies of D by D^m. For a set S, $x \leftarrow S$ denotes uniformly sampling from S. We denote the uniform distribution over S by $U\{S\}$.

We rely on the standard notions of Turing machines and Boolean circuits.

- We say that a Turing machine is PPT if it is probabilistic and runs in polynomial time.
- For a PPT algorithm M, we denote by $M(x; r)$ the output of M on input x and random coins r. For such an algorithm, and any input x, we may write $m \in M(x)$ to denote the fact that m is in the support of $M(x; \cdot)$.
- A polynomial-size circuit family $\mathcal{C}$ is a sequence of circuits $\mathcal{C} = \{C_\lambda\}_{\lambda \in \mathbb{N}}$, such that each circuit C_λ is of polynomial size $\lambda^{O(1)}$ and has $\lambda^{O(1)}$ input and output bits. We also consider probabilistic circuits that may toss random coins.

- We follow the standard convention of modeling any efficient adversary as a family of polynomial-size circuits. For an adversary A corresponding to a family of polynomial-size circuits $\{\mathsf{A}_\lambda\}_{\lambda\in\mathbb{N}}$, we sometimes omit the subscript λ, when it is clear from the context.

- A function $f : \mathbb{N} \to [0,1]$ is negligible if $f(\lambda) = \lambda^{-\omega(1)}$ and is noticeable if $f(\lambda) = \lambda^{-O(1)}$.

- Two ensembles of random variables $\mathcal{X} = \{X_i\}_{\lambda\in\mathbb{N}, i\in I_\lambda}$, $\mathcal{Y} = \{Y_i\}_{\lambda\in\mathbb{N}, i\in I_\lambda}$ over the same set of indices $I = \uplus_{\lambda\in\mathbb{N}} I_\lambda$ are said to be *computationally indistinguishable* (respectively, *statistically indistinguishable*), denoted by $\mathcal{X} \approx_c \mathcal{Y}$ (respectively, $\mathcal{X} \approx_s \mathcal{Y}$), if for every polynomial-size (respectively, unbounded) distinguisher $\mathsf{A} = \{\mathsf{A}_\lambda\}_{\lambda\in\mathbb{N}}$ there exists a negligible function μ such that for all $\lambda \in \mathbb{N}, i \in I_\lambda$,

$$|\Pr(\mathsf{A}(X_i) = 1) - \Pr(\mathsf{A}(Y_i) = 1)| \le \mu(\lambda) \ .$$

2.1 SSS Non Interactive Zero Knowledge Proofs

We define non-interactive zero knowledge with one-time statistical simulation soundness [Sah99]. Such NIZKs can be constructed from any NIZK proof and one-way functions (c.f. [GGH+13]), and in particular from IO and one-way functions [BP15].

Definition 2.1. *An SSS NIZK for an* NP *relation* $\mathcal{R}$ *consists of a triplet* $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ *where* Gen, P *are PPT and* V *is a deterministic polynomial-time algorithm with the following syntax:*

- $crs \leftarrow \mathsf{Gen}(1^\lambda)$ *is a generator that given a security parameter* 1^λ*, outputs a common reference string.*

- $\pi \leftarrow \mathsf{P}(crs, x, w)$ *is a prover prover that given a pair* $(x, w) \in \mathcal{R}$ *and reference string* crs*, outputs a proof* π*.*

- $b \leftarrow \mathsf{V}(crs, x, \pi)$ *is a verifier that given a statement* x*, a proof* π *and a reference string* crs*, outputs a single bit* b*.*

Let $\mathcal{L}$ *be the language implied by* $\mathcal{R}$ *(i.e.,* $\mathcal{L} := \{x | \exists w : (x, w) \in \mathcal{R}\}$*), we require the following properties:*

1. *Perfect Completeness: for any* $(x, w) \in \mathcal{R}$ *the prover convinces the verifier:*

$$\Pr\left[\mathsf{V}(crs, x, \pi) = 1 : crs \leftarrow \mathsf{Gen}(1^\lambda), \pi \leftarrow \mathsf{P}(crs, x, w)\right] = 1.$$

2. *Statistical Soundness: it's infeasible to convince an honest verifier of a false statement* $x \notin \mathcal{L}$*:*

$$\Pr\left[\exists(x, \pi)\ s.t.\ x \notin L \textit{ and } \mathsf{V}(crs, x, \pi) = 1 : crs \leftarrow \mathsf{Gen}(1^\lambda)\right] \le \mathrm{negl}(\lambda).$$

3. *Computational Zero Knowledge (ZK): the* crs *together with the proof* π *can be simulated. Formally, there exists a* poly*-time simulator* S *such:*

$$\left\{(crs, \pi) : \begin{array}{c} crs \leftarrow \mathsf{Gen}(1^\lambda) \\ \pi \leftarrow \mathsf{P}(crs, x, w) \end{array}\right\}_{\substack{\lambda\in\mathbb{N}, \\ (x,w)\in\mathcal{R}}} \approx_c \left\{(crs, \pi) \leftarrow \mathsf{S}(1^\lambda, x)\right\}_{\substack{\lambda\in\mathbb{N}, \\ (x,w)\in\mathcal{R}}}$$

4. *Statistical Simulation-Soundness (SSS): A* NIZK *protocol is said to be statistically simulation sound when it's not possible to convince an honest verifier of a false statement, even when given a simulated proof. Formally, for every* $x \in L$*:*

$$\Pr\left[\exists x' \notin L, \pi' \textit{and } V(crs, x', \pi') = 1 : \ (crs, \pi) \leftarrow \mathsf{S}(1^\lambda, x) \ \right] \le \mathrm{negl}(\lambda).$$

2.2 Indistinguishability Obfuscation

We define indistinguishability obfuscation (IO).

Definition 2.2 (Indistinguishability Obfuscation [BGI+01])**.** *Let $\mathcal{C} = \cup_{\lambda\in\mathbb{N}}\mathcal{C}_\lambda$ be a family of boolean circuits, where for every $\lambda \in \mathbb{N}$, $\mathcal{C}_\lambda$ is a set of poly-sized circuits. A PPT algorithm* iO *is an indistinguishability obfuscator for $\mathcal{C}$ if it satisfies:*

1. *Functionality: for every $\lambda \in \mathbb{N}$ and every circuit $C \in \mathcal{C}_\lambda$:*

$$Pr_{\mathsf{iO}}\left[\mathsf{iO}(C) \equiv C\right] = 1.$$

2. *Indistinguishability: For every pair of circuits $C_0, C_1 \in \mathcal{C}_\lambda$ of the same size and functionality, their obfuscations are indistinguishable. Formally:*

$$\{\mathsf{iO}(C_0)\}_{\substack{\lambda\in\mathbb{N},C_0,C_1\in\mathcal{C}_\lambda: \\ C_0\equiv C_1,|C_0|=|C_1|}} \approx_c \{\mathsf{iO}(C_1)\}_{\substack{\lambda\in\mathbb{N},C_0,C_1\in\mathcal{C}_\lambda: \\ C_0\equiv C_1,|C_0|=|C_1|}} .$$

2.3 Public Key Encryption

We rely on standard public-key encryption (PKE). For simplicity we stick to bit encryption (which is extended to multi-bit messages in the usual way).

Definition 2.3. *A public-key bit encryption scheme is a triplet of algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *where* Gen*,* Enc *are PPT and* Dec *is a deterministic polynomial-time with the following syntax:*

- $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ *is a generator that given the security parameter λ outputs the public and secret keys.*
- $ct \leftarrow \mathsf{Enc}_{pk}(m)$ *is an encryption algorithm that given a message $m \in \{0, 1\}$ and a public key pk outputs a ciphertext ct.*
- $m' \leftarrow \mathsf{Dec}_{sk}(ct)$ *is a decryption algorithm that given the secret key and a ciphertext ct outputs a message m'.*

We require the following properties:

1. *Correctness: For any message $m \in \{0, 1\}$:*

$$\Pr\left[\mathsf{Dec}_{sk}(ct) = m : \ (pk, sk) \leftarrow \mathsf{Gen}(1^\lambda), ct \leftarrow \mathsf{Enc}_{pk}(m) \ \right] = 1.$$

2. *Semantic Security: Encryptions of 0 and 1 are computationally indistinguishable:*

$$\left\{(pk, ct) : \begin{array}{l} (pk, sk) \leftarrow \mathsf{Gen}(1^\lambda) \\ ct \leftarrow \mathsf{Enc}_{pk}(0) \end{array} \right\}_{\lambda\in\mathbb{N}} \approx_c \left\{(pk, ct) : \begin{array}{l} (pk, sk) \leftarrow \mathsf{Gen}(1^\lambda) \\ ct \leftarrow \mathsf{Enc}_{pk}(1) \end{array} \right\}_{\lambda\in\mathbb{N}} .$$

3 Pseudo Non-Linear Codes

In this section we provide the definition of Pseudo Non-Linear Codes (PNLC), we then present two constructions of PNLC: the first from the Decisional Diffie-Hellman assumption and the second from the Learning With Errors assumption. This new primitive will be used later as a building block to construct robust Additive Randomized Encoding.

3.1 Pseudo Non-Linear Codes: Definition

Definition 3.1 (Pseudo-Non-Linear Coding (PNLC)). *A PNLC scheme is associated with a parameter $k = k(\lambda)$ and consists of four algorithms $\mathsf{PNLC} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Add}, \mathsf{Eq})$ where $\mathsf{Gen}, \mathsf{Enc}$ are are PPT algorithms and $\mathsf{Add}, \mathsf{Eq}$ are polynomial-time deterministic algorithms, with the following syntax:*

- $pp \leftarrow \mathsf{Gen}(1^\lambda)$ *is a generator that given the security parameter 1^λ, generates public parameters pp. The public parameters include the description of an Abelian group $(\mathbb{G}, +)$ with efficient representation and operations.*
- $\hat{s} \leftarrow \mathsf{Enc}_{pp}(s, i)$ *is an encoder that given an element $s \in \mathbb{G}$, and index $i \in [k] \cup \{+\}$, outputs an encoding $\hat{s}$.*
- $\hat{s} \leftarrow \mathsf{Add}_{pp}(\hat{\boldsymbol{s}})$ *is an addition procedure, that given a vector $\hat{\boldsymbol{s}}$ of k encodings, outputs a new encoding $\hat{s}$.*
- $b \leftarrow \mathsf{Eq}_{pp}(\hat{s}, \hat{t})$ *is an equality checker that given two encodings, outputs a bit $b \in \{0, 1\}$.*

We require the following properties:

1. *Additivity: The encodings are additive. Formally, for any $\boldsymbol{s} \in \mathbb{G}^k$, and letting $t = \sum_{i \in [k]} s_i$,*
$$\Pr\left[\mathsf{Eq}_{pp}\left(\mathsf{Add}_{pp}(\hat{\boldsymbol{s}}), \hat{t}\right) = 1\right] = 1.$$
where $pp \leftarrow \mathsf{Gen}(1^\lambda)$, $\hat{s}_i \leftarrow \mathsf{Enc}_{pp}(s_i, i)$ for all $i \in [k]$, and $\hat{t} \leftarrow \mathsf{Enc}_{pp}(t, +)$.
2. *Pseudo Non-Linearity: There exists a PPT simulator $\mathsf{Sim}(pp, H)$, which given public parameters pp and $H \subseteq [k]$, outputs a vector $\widetilde{\boldsymbol{\sigma}}_H$ of h fake encodings and a group element τ such that:*
   - *Indistinguishability: Fake encodings are computationally indistinguishable from real random encodings of some secrets together with their sum:*
   $$\{pp, \hat{\mathbf{s}}_H, t\}_{\substack{\lambda \in \mathbb{N}, \\ H \subseteq [k]}} \approx_c \{pp, \widetilde{\boldsymbol{\sigma}}_H, \tau\}_{\substack{\lambda \in \mathbb{N}, \\ H \subseteq [k]}}$$
   where $pp \leftarrow \mathsf{Gen}(1^\lambda)$, $s_i \leftarrow \mathbb{G}$, $\hat{s}_i \leftarrow \mathsf{Enc}_{pp}(s_i, i)$ for all $i \in H$, $t = \sum_{i \in H} s_i$, and $(\widetilde{\boldsymbol{\sigma}}_H, \tau) \leftarrow \mathsf{Sim}(pp, H)$.
   - *Non-Linearity: The sum of any strict subset of fake encodings evades the code: for any $H \subseteq [k]$,*
   $$\Pr\left[\exists \emptyset \subsetneq F \subsetneq H, \hat{\boldsymbol{s}}_{\overline{F}}, \hat{t} : \mathsf{Eq}_{pp}\left(\mathsf{Add}_{pp}(\widetilde{\boldsymbol{\sigma}}_F, \hat{\boldsymbol{s}}_{\overline{F}}), \hat{t}\right) = 1\right] \leq \operatorname{negl}(\lambda) \ ,$$
   where $pp \leftarrow \mathsf{Gen}(1^\lambda), (\widetilde{\boldsymbol{\sigma}}_H, \tau) \leftarrow \mathsf{Sim}(pp, H)$, $\hat{s}_i$ is a (valid) encoding in the support of $\mathsf{Enc}_{pp}(s_i, i)$ for some $s_i \in \mathbb{G}$ for all $i \in \overline{F}$, and $\hat{t}$ is a (valid) encoding in the support of $\mathsf{Enc}_{pp}(t, +)$ for some $t \in \mathbb{G}$.

Remark 3.1 (Inefficient simulation, and symmetric encodings). A possible relaxation to the definition is allowing the simulator to be unbounded. This would suffice for our needs (but would come at the cost of inherently non-uniform security reductions).

3.2 Construction from Decisional Diffie-Hellman

In this section we construct Pseudo Non-Linear Codes from the Decisional Diffie-Hellman (DDH) assumption.

DDH Groups in Additive Notation. It will be convenient to denote group encodings in additive notation (following e.g. [BHHO08]). We consider a prime-order DDH group $\mathbb{H}$ and identify it with $(\mathbb{Z}_p, +)$. A generator $g \in \mathbb{G}$ will be denote by $[1]$, every element g^x by $[x]$, and $g^x \times g^y$ by $[x] + [y] = [x + y]$. We may similarly consider vectors of encodings $[\boldsymbol{x}] = (x_1, \ldots, x_k)$. For two vectors $\boldsymbol{x}, \boldsymbol{y}$ of length k, we will abuse notation and denote by $([\boldsymbol{x}], [\boldsymbol{y}])$ the vector of pairs $(([x_1], [y_1]), \ldots, ([x_k], [y_k]))$.

Definition 3.2 (Decisional Diffie-Hellman)**.** *Let $\mathcal{G}$ be a PPT generaotr that given 1^λ outputs $(\mathbb{H}, p, [1])$, where $\mathbb{H}$ is a group of prime order p with generator $[1]$ (and efficient representation and operations). The DDH assumption with respect to $\mathcal{G}$ states that*

$$\left\{ \begin{array}{c} \mathbb{H}, p, [1] \\ [x], [y], [xy] \end{array} \right\}_{\lambda \in \mathbb{N}} \approx_c \left\{ \begin{array}{c} \mathbb{H}, p, [1] \\ [x], [y], [z] \end{array} \right\}_{\lambda \in \mathbb{N}} ,$$

where $(\mathbb{H}, p, [1]) \leftarrow \mathcal{G}(1^\lambda), x, y, z \leftarrow \mathbb{Z}_p$.

Remark 3.2 (Multi-Instance DDH). We will rely on the following variant, which follows from DDH by a standard hybrid argument:

$$\left\{ \begin{array}{c} \mathbb{H}, p, [1] \\ [x], [\boldsymbol{y}], [x\boldsymbol{y}] \end{array} \right\}_{\lambda \in \mathbb{N}} \approx_c \left\{ \begin{array}{c} \mathbb{H}, p, [1] \\ [x], [\boldsymbol{y}], [\boldsymbol{z}] \end{array} \right\}_{\lambda \in \mathbb{N}} ,$$

where $(\mathbb{H}, p, [1]) \leftarrow \mathcal{G}(1^\lambda), x \leftarrow \mathbb{Z}_p, \boldsymbol{y}, \boldsymbol{z} \leftarrow \mathbb{Z}_p^t$, where $t(\lambda)$ is any polynomial.

We now describe our DDH-based construction of PNLC. The construction is given in Figure 1. In this specific construction, the encoding algorithm will be deterministic and symmetric (independent of the index i), likewise, the encodings addition algorithm will be symmetric. Accordingly, we omit the index $i \in [k] \cup \{+\}$ from the encoding algorithm inputs.

PNLC Scheme

$\mathsf{Gen}(1^n)$**:**

- Sample $(\mathbb{H}, p, [1]) \leftarrow \mathcal{G}(1^\lambda)$, and $x \leftarrow \mathbb{Z}_p$.
- Output $pp = (\mathbb{G} = \mathbb{Z}_p, \mathbb{H}, [1], [x])$.

$\mathsf{Enc}_{pp}(s)$**:**

- Output $\hat{s} = ([s], [xs])$.

$\mathsf{Add}_{pp}(\hat{\boldsymbol{s}})$**:**

- Parse: $\hat{\boldsymbol{s}} = ([\boldsymbol{s}], [\boldsymbol{r}]) \in \mathbb{H}^{2 \times k}$.
- Output $\sum_{i=1}^{k} ([s_i], [r_i])$.

$\mathsf{Eq}_{pp}(\hat{s}, \hat{t})$**:**

- Output 1 iff $\hat{s} = \hat{t}$.

Figure 1: Pseudo Non-Linear Codes from Decisional Diffie-Hellman.

Proposition 3.3. *The construction is additive.*

Proof. The proof follows directly from the additivity of group encodings. Let $\boldsymbol{s} \in \mathbb{Z}_p^k, x \in \mathbb{Z}_p, pp = (\mathbb{G} := \mathbb{Z}_p, \mathbb{H}, [1], [x])$, and $t = \sum_{i=1}^{k} s_i$. Then

$$\mathsf{Add}_{pp}(\hat{\boldsymbol{s}}) = \mathsf{Add}_{pp}([\boldsymbol{s}], [x\boldsymbol{s}]) = \sum_{i=1}^{k} ([s_i], [xs_i]) = \left(\left[\sum_{i=1}^{k} s_i \right], \left[x \sum_{i=1}^{k} s_i \right] \right) = \hat{t} \ .$$

$\square$

Proposition 3.4. *The construction is pseudo non-linear.*

Proof. Let $H \subseteq [k]$ be of size h. We define $\mathsf{Sim}(pp, H)$, where pp include $([1],[x])$ to output

$$(\widetilde{\boldsymbol{\sigma}}_H, \tau) = \left(([\boldsymbol{s}],[\boldsymbol{r}]), \left(\left[\tau - \sum_{i=1}^{h-1} s_i\right], \left[x\tau - \sum_{i=1}^{h-1} r_i\right]\right)\right), \tau \quad \text{where} \quad \boldsymbol{s}, \boldsymbol{r} \leftarrow \mathbb{Z}_p^{h-1}, \tau \leftarrow \mathbb{Z}_p \ .$$

We next prove that the simulator satisfies the two properties required by pseudo non-linearity.

Indistinguishability. Assume toward contradiction there exists an efficient adversary distinguisher D that given $pp = (\mathbb{Z}_p, \mathbb{H}, [1], [x])$ distinguishes, the real distribution

$$[\boldsymbol{s}'], [x\boldsymbol{s}'], t = \sum_{i=1}^{h} s'_i \quad \text{where} \quad \boldsymbol{s}' \leftarrow \mathbb{Z}_p^h$$

from the simulated distribution defined above.

We use D to break DDH, and specifically to distinguish $[x], [\boldsymbol{s}], [x\boldsymbol{s}]$ from $[x], [\boldsymbol{s}], [\boldsymbol{r}]$ where $\boldsymbol{s}, \boldsymbol{r} \leftarrow \mathbb{Z}_p^{h-1}, x \leftarrow \mathbb{Z}_p$. Given $([x], [\boldsymbol{s}], [\boldsymbol{z}])$, we sample $\tau \leftarrow \mathbb{Z}_p$ and compute

$$([s_h], [z_h]) = \left(\left[\tau - \sum_{i=1}^{h-1} s_i\right], \left[\tau x - \sum_{i=1}^{h-1} z_i\right]\right) \ .$$

We then run D on the corresponding pp and $[\boldsymbol{s}|s_h], [\boldsymbol{z}|z_h], \tau$. Note that if $\boldsymbol{z}$ is distributed as $x\boldsymbol{s}$, then the resulting distribution is exactly the real distribution, whereas if $\boldsymbol{z}$ is distributed as a random $\boldsymbol{r}$, the distribution is exactly the simulated distribution. Hence D manages to distinguish with the exact same advantage.

Non-linearity. For any $H \subseteq [k]$ and any strict non-empty subset of coordinates $\emptyset \subsetneq F \subsetneq H$, the simulated values $\widetilde{\boldsymbol{\sigma}}_F = ([\boldsymbol{s}_F], [\boldsymbol{r}_F])$ are such that $(\boldsymbol{s}, \boldsymbol{r})$ is distributed uniformly at random over $(\mathbb{Z}_p \times \mathbb{Z}_p)^{|F|}$, and in particular $\sum_{i \in F} (\boldsymbol{s}, \boldsymbol{r})$ is distributed uniformly at random over $\mathbb{Z}_p \times \mathbb{Z}_p$. Furthermore, there exists valid encodings $\hat{\boldsymbol{s}}_{\overline{F}} = ([\boldsymbol{s}_{\overline{F}}], [x\boldsymbol{s}_{\overline{F}}]), \hat{t} = ([t], [xt])$ such that $\mathsf{Add}_{pp}(\widetilde{\boldsymbol{\sigma}}_F, \hat{\boldsymbol{s}}_{\overline{F}}) = \hat{t}$ iff $\sum_{i \in F} (s_i, r_i)$ is spanned by $(1, x)$, which occurs with probability $1/p = \mathrm{negl}(\lambda)$ over the choice of $\widetilde{\boldsymbol{\sigma}}_F$. $\square$

3.3 Construction from Learning With Errors

In this section we construct Pseudo Non-Linear Codes from the Learning with Errors (LWE) assumption. We start by recall the definition of LWE. Throughout this section, we follow the habit of identifying the security parameter with the LWE dimension n.

Definition 3.5 ((Decisional) LWE [Reg09]). *Let $n \in \mathbb{N}$ and $q = q(n) \geq 2$ an integer modulus, and let $\chi = \chi(n)$ be an error distribution over $\mathbb{Z}$. The $\mathsf{LWE}_{n,q,\chi}$ hardness assumption asserts that for any $m = \mathrm{poly}(n)$:*

$$\{\mathbf{A}, \mathbf{As} + \mathbf{e} \mod q\}_{n \in \mathbb{N}} \approx_c \{\mathbf{A}, \mathbf{u}\}_{n \in \mathbb{N}}$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \chi^m$, and $\mathbf{u} \leftarrow \mathbb{Z}_q^m$.

Remark 3.3 (Multi-Secret LWE). We will rely on the following variant, which follows from LWE by a standard hybrid argument:

$$\{\mathbf{A}, \mathbf{AS} + \mathbf{E} \mod q\}_{n \in \mathbb{N}} \approx_c \{\mathbf{A}, \mathbf{U}\}_{n \in \mathbb{N}} \ ,$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}$, $\mathbf{S} \leftarrow \mathbb{Z}_q^{n \times k}$, $\mathbf{E} \leftarrow \chi^{m \times k}$, $\mathbf{U} \leftarrow \mathbb{Z}_q^{m \times k}$ and $k(\lambda)$ is any polynomial.

Definition 3.6. *A distribution on $\mathbb{Z}$ is b-bounded if it is supported on $[-b, b]$.*

There are (quantum and classical) reductions between LWE and approximating short vector problems in lattices in the worst case, where χ is a b-bounded (truncated) Gaussian and the approximation parameter scales with q/b [Reg09, Pei09, BLP+13]. In known algorithms the corresponding lattice problems run in time $2^{\tilde{O}(n/k)}$ to obtain approximation 2^k [AKS01, MV10]. Accordingly, given known reductions, LWE is plausible for any $b = \mathrm{poly}(n) < q < 2^{n^{\varepsilon}}$ where $\varepsilon < 1$.

Lemma 3.7 (Smudging [AJLA+12])**.** *Let $B(n)$ and $b(n)$ be such that $b/B = \mathrm{negl}(n)$. Then for every $y \in [-b, b]$ and $x \leftarrow U[-B, B]$, the distributions x and $x + y$ are statistically indistinguishable.*

We now describe our LWE-based construction of PNLC. The construction is given in Figure 2. Like in the DDH construction, the encoding and addition algorithms will be symmetric (independent of the index i). Accordingly, we omit the index $i \in [k] \cup \{+\}$ from the encoding algorithm inputs. (However, unlike the DDH construction, the encoding will be randomized). In this scheme our PNLC group will be $\mathbb{G} = \mathbb{Z}_q^n$, consistently, we will describe vectors $(\mathbf{s}_1, \ldots, \mathbf{s}_k) \in \mathbb{G}^k$ as matrices $\mathbf{S} \in \mathbb{Z}_q^{n \times k}$, and for $I \subseteq [k]$ denote by $\mathbf{S}_I$ the columns corresponding to index set I. Throughout, all operations are done $\mod q$. For elements $z \in \mathbb{Z}_q$, $|z| \in \mathbb{Z}^+$ denotes the size of their representative in $[-q/2, q/2)$, and for a vector $\boldsymbol{z}$, we define $\|\boldsymbol{z}\|_\infty = \max_i |z_i|$.

PNLC Shceme

Parameters: $m(n) = 10n \log q$, $\quad n^{\omega(1)} \leq B(n) \leq q/5(k+1)$

$\mathsf{Gen}(1^n)$**:**

- Sample $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}$,
- Output $pp = \big(\mathbb{G} = \mathbb{Z}_q^n, \mathbf{A}\big)$.

$\mathsf{Enc}_{pp}(\mathbf{s})$**:**

- Sample $\mathbf{e} \leftarrow U[-B, B]^m$,
- Output $\hat{\mathbf{s}} := \mathbf{A}\mathbf{s} + \mathbf{e}$.

$\mathsf{Add}_{pp}(\widehat{\mathbf{S}})$**:**

- Output $\sum_{i=1}^{k} \hat{\mathbf{s}}_i$.

$\mathsf{Eq}_{pp}(\hat{\mathbf{s}}, \hat{\mathbf{t}})$**:**

- Output 1 iff for $\|\hat{\mathbf{s}} - \hat{\mathbf{t}}\|_\infty \leq q/5$.

Figure 2: Pseudo Non-Linear Codes from Learning With Errors.

Proposition 3.8. *The construction is additive.*

Proof. The proof follows directly from the linearity of the code given by $\mathbf{A}$. Let $\mathbf{S} \in \mathbb{Z}_q^{n \times k}, \mathbf{A} \in \mathbb{Z}_q^{m \times n}$, $pp = \big(\mathbb{G} := \mathbb{Z}_q^n, \mathbf{A}\big)$, $\mathbf{t} = \sum_{i=1}^{k} \mathbf{s}_i$, $\hat{\mathbf{s}}_i = \mathbf{A}\mathbf{s}_i + \mathbf{e}_i$, $\hat{\mathbf{t}} = \mathbf{A}\mathbf{t} + \mathbf{e}_+$. Then

$$\mathsf{Add}_{pp}(\widehat{\mathbf{S}}) = \sum_{i=1}^{k} \hat{\mathbf{s}}_i = \mathbf{A}\mathbf{t} + \sum_{i=1}^{k} \mathbf{e}_i \ ,$$

and thus

$$\left\|\hat{\mathbf{t}} - \sum_{i=1}^{k} \hat{\mathbf{s}}_i\right\|_\infty = \left\|\sum_{i=1}^{k} \hat{\mathbf{e}}_i - \mathbf{e}_+\right\|_\infty \leq (k+1)B \leq q/5 \ .$$

□

Proposition 3.9. *The construction is pseudo non-linear.*

Proof. Let $H \subseteq [k]$ be of size h. We define $\mathsf{Sim}(pp, H)$, where pp include $\mathbf{A}$ to output

$$(\widetilde{\boldsymbol{\sigma}}_H, \tau) = \left(\mathbf{U}, \mathbf{A}\boldsymbol{\tau} - \sum_{i=1}^{h-1} \mathbf{u}_i\right) + \mathbf{E}', \boldsymbol{\tau} \quad \text{ where } \quad \mathbf{U} \leftarrow \mathbb{Z}_q^{m\times(h-1)}, \boldsymbol{\tau} \leftarrow \mathbb{Z}_q^n, \mathbf{E}' \leftarrow U[-B, B]^{m\times h} \ .$$

We next prove that the simulator satisfies the two properties required by pseudo non-linearity.

Indistinguishability. Assume toward contradiction there exists an efficient adversary distinguisher D that given $pp = (\mathbb{Z}_q^n, \mathbf{A})$ distinguishes, the real distribution

$$\mathbf{AS} + \mathbf{E}, \mathbf{t} = \sum_{i=1}^{h} \mathbf{s}_i \quad \text{ where } \quad \mathbf{S} \leftarrow \mathbb{Z}_q^{n\times h}, \mathbf{E} \leftarrow U[-B, B]^{m\times h}$$

from the simulated distribution defined above.

We use D to break $\mathsf{LWE}_{n,q,\chi}$ for any b-bounded χ such that $b/B = \mathrm{negl}(n)$. Specifically we distinguish $(\mathbf{A}, \mathbf{AS}+\mathbf{E})$ from $(\mathbf{A}, \mathbf{U})$ where $\mathbf{A} \leftarrow \mathbb{Z}_q^{m\times n}, \mathbf{S} \leftarrow \mathbb{Z}_q^{n\times(h-1)}, \mathbf{E} \leftarrow \chi^{m\times(h-1)}$. Given $(\mathbf{A}, \mathbf{B})$, we sample $\boldsymbol{\tau} \leftarrow \mathbb{Z}_q^n$ and compute $\boldsymbol{b}_h = \mathbf{A}\boldsymbol{\tau} - \sum_{i=1}^{h-1} \boldsymbol{b}_i$. We then sample $\mathbf{E}' \leftarrow U[-B, B]^{m\times h}$ and run D on the corresponding pp and $(\mathbf{B}, \boldsymbol{b}_h) + \mathbf{E}', \boldsymbol{\tau}$.

Note that if $\mathbf{B}$ is distributed as $\mathbf{AS} + \mathbf{E}$, then the resulting distribution is exactly the real distribution, except that the corresponding noise is

$$\left(\mathbf{E}, -\sum_{i=1}^{h-1} \mathbf{e}_i\right) + \mathbf{E}' \ ,$$

instead of just $\mathbf{E}'$. However, since $\|\mathbf{E}\|_\infty \leq b$,

$$\left\|\left(\mathbf{E}, -\sum_{i=1}^{h-1} \mathbf{e}_i\right)\right\|_\infty \leq (h-1)b \ ,$$

it follows by smudging (Lemma 3.7) that the distribution is statistically close to the real one.

In contrast, if $\mathbf{B}$ is distributed as $\mathbf{U}$, the distribution is exactly the simulated distribution. Hence D manages to distinguish with a negligibly close advantage to its original distinguishing advantage.

Non-linearity. For any $H \subseteq [k]$ and any strict non-empty subset of coordinates $\emptyset \subsetneq F \subsetneq H$, the simulated values $\widetilde{\boldsymbol{\sigma}}_F = \mathbf{B}_F$ are such that $\mathbf{B}$ is distributed uniformly at random over $\mathbb{Z}_q^{n\times|F|}$, and in particular $\sum_{i\in F} \boldsymbol{b}_i$ is distributed uniformly at random over $\mathbb{Z}_q^n$. Furthermore, there exists valid encodings $\hat{\mathbf{S}}_{\overline{F}} = \mathbf{AS} + \mathbf{E}, \hat{\mathbf{t}} = \mathbf{At} + \mathbf{e}$ such that $\mathsf{Add}_{pp}(\widetilde{\boldsymbol{\sigma}}_F, \widehat{\mathbf{S}}_{\overline{F}}) = \hat{\mathbf{t}} + \mathbf{e}'$ for $\|\mathbf{e}'\|_\infty \leq q/5$ iff

$$\sum_{i\in F} \boldsymbol{b}_i = \mathbf{A}\left(\mathbf{t} - \sum_{i\notin F} \mathbf{s}_i\right) + \mathbf{e} - \sum_{i\in F} \mathbf{e}_i + \mathbf{e}' \ ,$$

where

$$\left\|\mathbf{e} - \sum_{i\in F} \mathbf{e}_i + \mathbf{e}'\right\|_\infty \leq (|F|+1)B + q/5 \leq 2q/5 \ .$$

By a union bound over the image of $\mathbf{A}$ and all possible errors as above, this occurs with probability at most

$$|\mathbb{Z}_q^n| \cdot (4/5)^m \leq \mathrm{negl}(n) \ ,$$

for our choice $m = 10n \log q$. □

Remark 3.4 (Parameters). A reasonable choice of parameters would be for instance $b = \mathrm{poly}(n)$, $q = n^{\log n}$, $B \approx \sqrt{q}$.

4 Robust Additive Randomized Encoding: Definition and Construction

Halevi, Ishai, Kushilevitz, and Rabin [HIKR23] define the notion of an additive randomized encoding (ARE) as well as robust additive randomized encoding (RARE), which we focus on here. Next we define the notion, and then show a construction from IO and PNLC.

4.1 Definition

We now define the notion of RARE.

Definition 4.1 (Robust Additive Randomized Encoding (RARE) [HIKR23])**.** *A RARE scheme for a function* $f : \{0,1\}^{n\times k} \to \{0,1\}^*$, *for polynomially bounded functions* $n(\lambda), k(\lambda)$ *consists of three PPT algorithms* $\mathsf{RARE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ *with the following syntax:*

1. $pp \leftarrow \mathsf{Setup}(1^\lambda)$ *is a generator that given the security parameter* 1^λ *generates public parameters pp. The public parameters include the description of an Abelian group* $(\mathbb{G}, +)$ *with efficient representation and operations.*
2. $(\hat{z}, \hat{g}) \leftarrow \mathsf{Enc}_{pp}(x, i)$ *is an encoding algorithm that given an input* $x \in \{0,1\}^n$ *and an index* $i \in [k]$ *outputs an encoding pair* $\hat{z} \in \{0,1\}^*, \hat{g} \in \mathbb{G}$.
3. $y \leftarrow \mathsf{Dec}_{pp}(\hat{\boldsymbol{z}}, \hat{g})$ *is a decoding algorithm that given* $\hat{\boldsymbol{z}} \in (\{0,1\}^*)^k, \hat{g} \in \mathbb{G}$ *outputs a value* y.

We require the following properties:

1. *Correctness:* RARE *is (perfectly) correct if for all* λ *and* $\boldsymbol{x} \in (\{0,1\}^n)^k$*:*

$$\Pr\left[\mathsf{Dec}_{pp}(\hat{\boldsymbol{z}}, \hat{g}) = f(\boldsymbol{x}) : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda) \\ (\hat{z}_i, \hat{g}_i) \leftarrow \mathsf{Enc}_{pp}(x_i, i) \\ \hat{g} = \sum_{i=1}^k \hat{g}_i \end{array}\right] = 1 \ .$$

2. *Indistinguishability Robustness: For any set of parties* $H \subseteq [k]$ *the partial encoding* $(\boldsymbol{z}_H, \sum_{i\in H} \hat{g}_i)$ *only reveals the residual function* $f_{H,\boldsymbol{x}}(\boldsymbol{y}) = f(\boldsymbol{x}_H, \boldsymbol{y}_{\bar{H}})$*. Formally,*

$$\left\{pp, \boldsymbol{z}_H, \sum_{i\in H} \hat{g}_i\right\}_{\substack{\lambda\in\mathbb{N}, H\subseteq[k] \\ \boldsymbol{x},\boldsymbol{x}' : f_{H,\boldsymbol{x}} \equiv f_{H,\boldsymbol{x}'}}} \approx_c \left\{pp, \boldsymbol{z}'_H, \sum_{i\in H} \hat{g}'_i\right\}_{\substack{\lambda\in\mathbb{N}, H\subseteq[k] \\ \boldsymbol{x},\boldsymbol{x}' : f_{H,\boldsymbol{x}} \equiv f_{H,\boldsymbol{x}'}}} ,$$

where $pp \leftarrow \mathsf{Setup}(1^\lambda)$*,* $(\hat{z}_i, \hat{g}_i) \leftarrow \mathsf{Enc}_{pp}(x_i, i)$*, and* $(\hat{z}'_i, \hat{g}'_i) \leftarrow \mathsf{Enc}_{pp}(x'_i, i)$ *for all* $i \in H$.[2]

[2] As noted in [HIKR23] an equivalent formulation requires the existence of an unbounded simulator that can computationally simulate the above distributions given oracle access to $f_{H,\boldsymbol{x}}$.

Online and Offline Encodings. We note that our definition slightly generalizes the definition of [HIKR23] by decomposing the encoding of any input x_i into two parts $\hat{z}_i$ and $\hat{g}_i$, where we view the parts $\hat{z}_i$ as *offline encodings*, which can be communicated directly to the decoder, and we view the parts $\hat{g}_i$ as *online encodings*, which need to be summed together before being sent to the decoder.

The definition of [HIKR23] considers single part encodings $\hat{x}_i$, which are all summed together. We note that this notion follows directly from our notion as we can always extend $\mathbb{G}$ into a larger group $\mathbb{G} \times \mathbb{H}^k$ where we can represent $\hat{z}_i$ in $\mathbb{H}$ and set $\hat{x}_i = (0 \dots 0, \hat{z}_i, 0 \dots 0, \hat{g}_i)$, where $\hat{z}_i$ is embedded in the ith coordinate. The benefit in the decomposition is that we can explicitly aim to minimize the *online part*, which is conceptually the expensive part, which requires the parties to communicate to compute the sum, or some other means (e.g. shuffling via anonymous communication as discussed in the introduction).

We say that the a RARE is *online succinct* if the size of the additive group $\mathbb{G}$ only depends on the security parameter, and in particular does not depend on the input size n, the number of parties k, nor the circuit size $|f|$.

Definition 4.2 (Online Succinctness)**.** *A RARE is online succinct if* $\log |\mathbb{G}| \leq \lambda$.

4.2 Construction from IO and PNLC

We now provide our RARE construction.

Primitives:

1. A public key encryption scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.
2. An indistinguishability obfuscator iO for general circuits.
3. A statistical-simulation-soundness non-interactive zero-knowledge proof $\mathsf{NIZK} = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$.
4. A pseudo non-linear code $\mathsf{PNLC} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Add}, \mathsf{Eq})$.

We next describe our RARE for a function $f : (\{0,1\}^n)^k \to \{0,1\}^*$, represented by a circuit, where $k(\lambda)$ indicates the number of parties, $n(\lambda)$ indicates the input length, and $\lambda \in \mathbb{N}$ is the security parameter. The construction is formally given in Figure 3.

The Decoding Program D. We now define the decoding program

$$\mathsf{D}[pp, \boldsymbol{crs}, \boldsymbol{pkl}, \boldsymbol{pkr}, \boldsymbol{skl}](\boldsymbol{ctl}, \boldsymbol{ctr}, \boldsymbol{\pi}, t) \ ,$$

with hardwired values $pp, \boldsymbol{crs}, \boldsymbol{pkl}, \boldsymbol{pkr}, \boldsymbol{skl}$ and input $\boldsymbol{ctl}, \boldsymbol{ctr}, \boldsymbol{\pi}, t$. The program D is formally described in Figure 4.

Theorem 4.3. *The construction given in Figure 3 is a Robust ARE. Furthermore, it is online succinct.*

Proof. The correctness follows readily from the correctness of PNLC and IO. Online succinctness also follows readily from the construction of the RARE encoding algorithm. In what follows we prove security. Fix any set $H \subseteq [k]$ of honest parties with input $\boldsymbol{x}^*_H$, where for each $i \in H$, $x^*_i \in \{0,1\}^n$. We consider the distribution:

$$\left(PP, \boldsymbol{z}_H, \sum_{i \in H} \hat{g}_i\right) = \left((pp, \boldsymbol{crs}, \boldsymbol{pkl}, \boldsymbol{pkr}, \widetilde{\mathsf{D}}[pp, \boldsymbol{crs}, \boldsymbol{pkl}, \boldsymbol{pkr}, \boldsymbol{skl}]), (\boldsymbol{ctl}^*_H, \boldsymbol{ctr}^*_H, \boldsymbol{\pi}_H), \sum_{i \in H} s^*_i\right) \ ,$$

where $PP \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\hat{z}_i, \hat{g}_i) \leftarrow \mathsf{Enc}_{PP}(x_i, i)$ for all $i \in H$. We prove by a hybrid argument that this distribution is computationally indistinguishable from the same distribution where $\boldsymbol{x}^*_H$ is replaced with any $\boldsymbol{x'}_H$ such that $f_{H,\boldsymbol{x}^*_H} \equiv f_{H,\boldsymbol{x'}_H}$.

Hybrid 0**:** This is the real experiment capturing the above distribution.

RARE Scheme

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

1. Generate $pp \leftarrow \mathsf{PNLC.Gen}(1^\lambda)$.
2. For every party $i \in [k]$:
   (a) Generate a pair of left and right keys: $(pkl_i, skl_i), (pkr_i, skr_i) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$,
   (b) Generate a common reference string $crs_i \leftarrow \mathsf{NIZK.Gen}(1^\lambda)$.
3. Sample $\widetilde{\mathsf{D}} = \mathsf{iO}(\mathsf{D}[pp, \boldsymbol{crs}, \boldsymbol{pkl}, \boldsymbol{pkr}, \boldsymbol{skl}])$, where D is the decoding circuit given in Figure 4 padded to the maximal size among $\mathsf{D}, \mathsf{D}_3, \mathsf{D}_6$, where the last two are defined in the analysis.
4. Output $PP = (pp, \boldsymbol{crs}, \boldsymbol{pkl}, \boldsymbol{pkr}, \widetilde{\mathsf{D}})$. The group $\mathbb{G}$ is the one fixed by the PNLC public parameters pp.

$\underline{\mathsf{Enc}_{PP}(x_i, i)}$:

1. Sample $s_i \leftarrow \mathbb{G}$ and randomness $r_i \leftarrow \{0,1\}^\lambda$ for the PNLC encoding algorithm.
2. Sample $ctl_i \leftarrow \mathsf{Enc}_{pkl_i}(x_i, s_i, r_i)$ and $ctr_i \leftarrow \mathsf{Enc}_{pkr_i}(x_i, s_i, r_i)$.
3. Prepare a NIZK proof $\pi_i \leftarrow \mathsf{NIZK.P}(crs_i, (ctl_i, pkl_i, ctr_i, pkr_i), w_i)$ attesting that (ctl_i, ctr_i) encrypt under (pkl_i, pkr_i) the same plaintext.
4. Output $(\hat{z}_i, \hat{g}_i) = ((ctl_i, ctr_i, \pi_i), s_i)$.

$\underline{\mathsf{Dec}_{PP}(\hat{\boldsymbol{z}}, t)}$:

1. Parse $\hat{\boldsymbol{z}} = (\boldsymbol{ctl}, \boldsymbol{ctr}, \boldsymbol{\pi})$.
2. Output $\widetilde{D}(\boldsymbol{ctl}, \boldsymbol{ctr}, \boldsymbol{\pi}, t)$.

Figure 3: Robust Additive Randomized Encoding from IO and PNLC

Hybrid 1**:** In this hybrid we replace the CRS and proofs $(\boldsymbol{crs}_H, \boldsymbol{\pi}_H)$ with simulated ones $(\widetilde{\boldsymbol{crs}}_H, \widetilde{\boldsymbol{\pi}}_H)$, by applying $(\widetilde{crs}_i, \widetilde{\pi}_i) \leftarrow \mathsf{S}(1^\lambda, (ctl_i^*, pkl_i, ctr_i^*, pkr_i))$ for all $i \in H$. We obtain the distribution:

$$\left((pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \widetilde{\mathsf{D}}[pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \boldsymbol{skl}]), (\boldsymbol{ctl}^*_H, \boldsymbol{ctr}^*_H, \widetilde{\boldsymbol{\pi}}_H), \sum_{i \in H} s_i^* \right) ,$$

Indistinguishability follows from the zero knowledge property of NIZK.

Hybrid 2**:** In this hybrid we replace the right ciphertexts $\boldsymbol{ctr}^*_H$ encrypting $(\boldsymbol{x}^*_H, \boldsymbol{s}^*_H, \boldsymbol{r}^*_H)$ with ciphertexts $\widetilde{\boldsymbol{ctr}}^*_H$ encrypting $(\mathbf{0}, \mathbf{0}, \mathbf{0})$. We obtain the distribution:

$$\left((pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \widetilde{\mathsf{D}}[pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \boldsymbol{skl}]), (\boldsymbol{ctl}^*_H, \widetilde{\boldsymbol{ctr}}^*_H, \widetilde{\boldsymbol{\pi}}_H), \sum_{i \in H} s_i^* \right) ,$$

Indistinguishability follows from the security of PKE, where we rely on the fact that $\widetilde{\boldsymbol{\pi}}$ is simulated only from the ciphertexts themselves (independently of the underlying randomness).

Program D

Hardwired:

- pp, public parameters for pseudo-non-linear code,
- $\boldsymbol{crs}$, a vector of k common reference strings,
- $\boldsymbol{pkl}, \boldsymbol{pkr}$, two vectors of k public encryption keys,
- $\boldsymbol{skl}$, a vector of k decryption keys.

Input:

- $\boldsymbol{ctl}$, a vector of k (left) ciphertexts,
- $\boldsymbol{ctr}$, a vector of k (right) ciphertexts,
- $\boldsymbol{\pi}$, a vector of k proofs,
- t, a group element.

Computation:

1. For $i \in [k]$:
   (a) **Verify:** Apply $\mathsf{NIZK.V}(crs_i, (ctl_i, pkl_i, ctr_i, pkr_i), \pi_i)$ to verify that (ctl_i, ctr_i) are valid encryptions of the same plaintext. If the verifier rejects, abort and output $\perp$.
   (b) **Decrypt left:** Compute $(x_i, s_i, r_i) = \mathsf{PKE.Dec}_{skl_i}(ctl_i)$.
   (c) **Encode:** Compute $\hat{s}_i := \mathsf{PNLC.Enc}_{pp}(s_i, i; r_i)$.
2. **Check sum:** Compute $\hat{t} = \mathsf{PNLC.Enc}_{pp}(t, +)$ (with some canonical randomness) and apply $\mathsf{PNLC.Eq}_{pp}(\mathsf{Add}_{pp}(\hat{\boldsymbol{s}}), \hat{t})$ to verify that $\hat{t}$ encodes the sum of $s_1, \ldots, s_k$. If not abort and output $\perp$.
3. Output $f(\boldsymbol{x})$.

Figure 4: The decoding program D

Hybrid 3**:** In this hybrid we replace the program $\mathsf{D}[pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \boldsymbol{skl}]$ with the program:

$$\mathsf{D}_3[pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \boldsymbol{skr}, \boldsymbol{ctl}^*_H, \widetilde{\boldsymbol{ctr}}^*_H, \boldsymbol{x}^*_H, \widehat{\boldsymbol{s}}^*_H] \ .$$

The program D_3 is formally described in Figure 5.

Program D_3

Hardwired:

- pp, public parameters for pseudo-non-linear code,
- $(\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}})$, a vector of k common reference strings,
- $\boldsymbol{pkl}, \boldsymbol{pkr}$, two vectors of k public encryption keys,
- $\boldsymbol{skr}$, a vector of k (right) decryption keys,
- $(\boldsymbol{ctl}^*_H, \widetilde{\boldsymbol{ctr}^*_H})$, the vector of (left and right) ciphertexts, where $\boldsymbol{ctl}^*_H$ encrypts $(\boldsymbol{x}^*_H, \boldsymbol{s}^*_H, \boldsymbol{r}^*_H)$,
- $\boldsymbol{x}^*_H$, the vector of inputs,
- $\widehat{\boldsymbol{s}}^*_H$, the encodings corresponding to $(\boldsymbol{s}^*_H, \boldsymbol{r}^*_H)$, namely $\{\widehat{s}^*_i = \mathsf{PNLC.Enc}_{pp}(s^*_i, i; r^*_i)\}_{i \in H}$

Input:

- $\boldsymbol{ctl}$, a vector of k (left) ciphertexts,
- $\boldsymbol{ctr}$, a vector of k (right) ciphertexts,
- $\boldsymbol{\pi}$, a vector of k proofs,
- t, a group element.

Computation:

1. For $i \in [k]$:
   (a) **Verify:** Apply $\mathsf{NIZK.V}(crs_i, (ctl_i, pkl_i, ctr_i, pkr_i), \pi_i)$. If V rejects, abort and output $\perp$.

   If $i \in H$ and $(ctl_i, ctr_i) = (ctl^*_i, \widetilde{ctr^*_i})$:

   (b) $x_i = x^*_i$,
   (c) $\hat{s}_i = \hat{s}^*_i$,

   Otherwise:

   (b) **Decrypt right:** Compute $(x_i, s_i, r_i) = \mathsf{PKE.Dec}_{skr_i}(ctr_i)$.
   (c) **Encode:** Compute $\hat{s}_i = \mathsf{PNLC.Enc}_{pp}(s_i, i; r_i)$.
2. **Check sum:** Compute $\hat{t} = \mathsf{PNLC.Enc}_{pp}(t, +)$ (with some canonical randomness) and apply $\mathsf{PNLC.Eq}_{pp}(\mathsf{Add}_{pp}(\hat{\boldsymbol{s}}), \hat{t})$. If it rejects, abort and output $\perp$.
3. Output $f(\boldsymbol{x})$.

Figure 5: The decoding program D_3

We obtain the distribution:

$$\Big((pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \widetilde{\mathsf{D}}_3[pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \boldsymbol{skr}, \boldsymbol{ctl}^*_H, \widetilde{\boldsymbol{ctr}}^*_H, \boldsymbol{x}^*_H, \widehat{\boldsymbol{s}}^*_H]),$$
$$(\boldsymbol{ctl}^*_H, \widetilde{\boldsymbol{ctr}}^*_H, \widetilde{\boldsymbol{\pi}}_H), \sum_{i \in H} s^*_i\Big) \enspace ,$$

To show indistinguishability, we prove that with overwhelming probability the new obfuscated program D_3 computes the same function as D, and hence indistinguishability follow from the security of iO.

Claim 4.4. *Except with negligible probability over the choice of*

$$pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{skl}, \boldsymbol{pkr}, \boldsymbol{skr}, \boldsymbol{ctl}^*_H, \widetilde{\boldsymbol{ctr}}^*_H, \boldsymbol{s}^*_H, \boldsymbol{r}^*_H \enspace ,$$

the programs are functionally equivalent:

$$\mathsf{D}[pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \boldsymbol{skl}] \equiv \mathsf{D}_3[pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \boldsymbol{skr}, \boldsymbol{ctl}^*_H, \widetilde{\boldsymbol{ctr}}^*_H, \boldsymbol{x}^*_H, \widehat{\boldsymbol{s}}^*_H] \enspace .$$

Proof. We address the two differences in the programs:

1. On inputs where $(ctl_i, ctr_i) = (ctl^*_i, \widetilde{ctr}^*_i)$ for $i \in H$, when the corresponding proof π_i is accepted, the program D decrypts ctl^*_i, and uses the result (x^*_i, s^*_i, r^*_i) to compute itself $x_i = x^*_i$ and the encoding $\hat{s}_i = \hat{s}^*_i$. The program D_3 does not decrypt and compute, but has the same values $(x^*_i, \hat{s}^*_i)$ hardwired. Hence the functionality is not affected.
2. On inputs where $(ctl_i, ctr_i) \neq (ctl^*_i, \widetilde{ctr}^*_i)$ for $i \in H$, when the corresponding proof π_i is accepted, then D decrypts the left ciphertext, whereas D_3 decrypts the right ciphertext. However, from the statistical simulation soundness of NIZK, it holds with overwhelming probability over $\widetilde{crs}_i$ that for all such (ctl_i, ctr_i), the two encrypt the same. Hence the functionality is not affected.

$\square$

Hybrid 4: In this hybrid we replace the left ciphertexts $\boldsymbol{ctl}^*_H$ encrypting $(\boldsymbol{x}^*_H, \boldsymbol{s}^*_H, \boldsymbol{r}^*_H)$ with ciphertexts $\widetilde{\boldsymbol{ctl}}^*_H$ encrypting $(\boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0})$. We obtain the distribution:

$$\Big((pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \widetilde{\mathsf{D}}_3[pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \boldsymbol{skr}, \widetilde{\boldsymbol{ctl}}^*_H, \widetilde{\boldsymbol{ctr}}^*_H, \boldsymbol{x}^*_H, \widehat{\boldsymbol{s}}^*_H]),$$
$$(\widetilde{\boldsymbol{ctl}}^*_H, \widetilde{\boldsymbol{ctr}}^*_H, \widetilde{\boldsymbol{\pi}}_H), \sum_{i \in H} s^*_i\Big) \enspace ,$$

Indistinguishability follows from the security of PKE, where we rely on the fact that in this hybrid the adversary view is independent of the left secret keys skl_H.

Hybrid 5: In this hybrid we replace the PNLC encodings $\widehat{\boldsymbol{s}}^*_H$ and the sum $t = \sum_{i \in H} s^*_i$ with simulated ones $(\widetilde{\boldsymbol{\sigma}}_H, \tau) \leftarrow \mathsf{PNLC.Sim}(pp, H)$. We obtain the distribution:

$$\Big((pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \widetilde{\mathsf{D}}_3[pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \boldsymbol{skr}, \widetilde{\boldsymbol{ctl}}^*_H, \widetilde{\boldsymbol{ctr}}^*_H, \boldsymbol{x}^*_H, \widetilde{\boldsymbol{\sigma}}_H]),$$
$$(\widetilde{\boldsymbol{ctl}}^*_H, \widetilde{\boldsymbol{ctr}}^*_H, \widetilde{\boldsymbol{\pi}}_H), \tau\Big) \enspace ,$$

Indistinguishability follows from the security of PNLC.

Program D_6

Hardwired:

- pp, public parameters for pseudo-non-linear code,
- $(\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}})$, a vector of k common reference strings,
- $\boldsymbol{pkl}, \boldsymbol{pkr}$, two vectors of k public encryption keys,
- $\boldsymbol{skr}$, a vector of k (right) decryption keys,
- $(\widetilde{\boldsymbol{ctl}^*_H}, \widetilde{\boldsymbol{ctr}^*_H})$, the vector of (left and right) ciphertexts.
- $f_{H,\boldsymbol{x}^*_H}$, a circuit realizing the residual function $f_{H,\boldsymbol{x}^*_H}(\boldsymbol{x}_{\overline{H}})$,
- $\widetilde{\boldsymbol{\sigma}}_H$, the simulated encodings.

Input:

- $\boldsymbol{ctl}$, a vector of k (left) ciphertexts,
- $\boldsymbol{ctr}$, a vector of k (right) ciphertexts,
- $\boldsymbol{\pi}$, a vector of k proofs,
- t, a group element.

Computation:

1. For $i \in [k]$:

   (a) **Verify:** Apply $\mathsf{NIZK.V}(crs_i, (ctl_i, pkl_i, ctr_i, pkr_i), \pi_i)$. If V rejects, abort and output $\perp$.

   If $i \in H$ and $(ctl_i, ctr_i) = (\widetilde{ctl^*_i}, \widetilde{ctr^*_i})$:

   (b) $\cancel{x_i = x^*_i}$,

   (c) $\hat{s}_i = \widetilde{\sigma}_i$,

   Otherwise:

   (b) **Decrypt right:** Compute $(x_i, s_i, r_i) = \mathsf{PKE.Dec}_{skr_i}(ctr_i)$.

   (c) **Encode:** Compute $\hat{s}_i = \mathsf{PNLC.Enc}_{pp}(s_i, i; r_i)$.

2. **Check sum:** Compute $\hat{t} = \mathsf{PNLC.Enc}_{pp}(t, +)$ (with some canonical randomness) and apply $\mathsf{PNLC.Eq}_{pp}(\mathsf{Add}_{pp}(\hat{\boldsymbol{s}}), \hat{t})$. If it rejects, abort and output $\perp$.

3. If for all $i \in H$, $(ctl_i, ctr_i) = (\widetilde{ctl^*_i}, \widetilde{ctr^*_i})$, output $f_{H,\boldsymbol{x}^*_H}(\boldsymbol{x}_{\overline{H}})$.

4. Output $f(\boldsymbol{x})$.

Figure 6: The decoding program D_6

Hybrid 6**:** In this hybrid we replace the program $\mathsf{D}_3[pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \boldsymbol{skr}, \widetilde{\boldsymbol{ctl}^*_H}, \widetilde{\boldsymbol{ctr}^*_H}, \boldsymbol{x}^*_H, \widetilde{\boldsymbol{\sigma}}_H]$ with the program:

$$\mathsf{D}_6[pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \boldsymbol{skr}, \widetilde{\boldsymbol{ctl}^*_H}, \widetilde{\boldsymbol{ctr}^*_H}, f_{H,\boldsymbol{x}^*_H}, \widetilde{\boldsymbol{\sigma}}_H] \ .$$

The program D_6 is formally described in Figure 6.

We obtain the distribution:

$$\Big((pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \widetilde{\mathsf{D}}_6[pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \boldsymbol{skr}, \widetilde{\boldsymbol{ctl}^*_H}, \widetilde{\boldsymbol{ctr}^*_H}, f_{H,\boldsymbol{x}^*_H}, \widetilde{\boldsymbol{\sigma}}_H]),$$
$$(\widetilde{\boldsymbol{ctl}^*_H}, \widetilde{\boldsymbol{ctr}^*_H}, \widetilde{\boldsymbol{\pi}}_H), \tau \Big) \ ,$$

To show indistinguishability, we prove that with overwhelming probability the new obfuscated program D_6 computes the same function as D_3, and hence indistinguishability follow from the security of iO.

Claim 4.5. *Except with negligible probability over the choice of*

$$pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \boldsymbol{skr}, \widetilde{\boldsymbol{ctl}^*_H}, \widetilde{\boldsymbol{ctr}^*_H}, \widetilde{\boldsymbol{\sigma}}_H \ ,$$

the programs are functionally equivalent:

$$\mathsf{D}_3[pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \boldsymbol{skr}, \widetilde{\boldsymbol{ctl}^*_H}, \widetilde{\boldsymbol{ctr}^*_H}, \boldsymbol{x}^*_H, \widetilde{\boldsymbol{\sigma}}_H] \equiv$$
$$\mathsf{D}_6[pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \boldsymbol{skr}, \widetilde{\boldsymbol{ctl}^*_H}, \widetilde{\boldsymbol{ctr}^*_H}, f_{H,\boldsymbol{x}^*_H}, \widetilde{\boldsymbol{\sigma}}_H] \ .$$

Proof. By their definition, the two programs may only differ on inputs $(\boldsymbol{ctl}, \boldsymbol{ctr}, \boldsymbol{\pi}, t)$ such that for some $\emptyset \subsetneq F \subsetneq H$, it holds that

$$(\boldsymbol{ctl}_F, \boldsymbol{ctr}_F) = (\widetilde{\boldsymbol{ctl}}^*_F, \widetilde{\boldsymbol{ctr}}^*_F) \qquad (\boldsymbol{ctl}_{H\setminus F}, \boldsymbol{ctr}_{H\setminus F}) \neq (\widetilde{\boldsymbol{ctl}}^*_{H\setminus F}, \widetilde{\boldsymbol{ctr}}^*_{H\setminus F}) \ .$$

As in D_6 the values $\boldsymbol{x}_F$ are no longer set to $\boldsymbol{x}^*_F$ as in D_3. However, by pseudo-non-linearity of PNLC, with overwhelming probability over the choice of $\widetilde{\boldsymbol{\sigma}}_H$, both programs abort in the above case during the **Check sum** step. Indeed, it is guaranteed that for any such F and any valid encodings $\hat{s}_{\overline{F}}, \hat{t}$, with respect to indices $\overline{F}$ and $+$, respectively, it holds that

$$\mathsf{Eq}_{pp}\left(\mathsf{Add}_{pp}\left(\widetilde{\boldsymbol{\sigma}}_F, \hat{\boldsymbol{s}}_{\overline{F}}\right), \hat{t}\right) = 0.$$

□

Hybrid 7**:** In this hybrid we replace the circuit $f_{H,\boldsymbol{x}^*_H}$ with a circuit $f_{H,\boldsymbol{x}'_H}$, such that $f_{H,\boldsymbol{x}'_H} \equiv f_{H,\boldsymbol{x}^*_H}$. We obtain the distribution:

$$\Big((pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \widetilde{\mathsf{D}}_6[pp, (\widetilde{\boldsymbol{crs}}_H, \boldsymbol{crs}_{\overline{H}}), \boldsymbol{pkl}, \boldsymbol{pkr}, \boldsymbol{skr}, \widetilde{\boldsymbol{ctl}^*_H}, \widetilde{\boldsymbol{ctr}^*_H}, f_{H,\boldsymbol{x}'_H}, \widetilde{\boldsymbol{\sigma}}_H]),$$
$$(\widetilde{\boldsymbol{ctl}^*_H}, \widetilde{\boldsymbol{ctr}^*_H}, \widetilde{\boldsymbol{\pi}}_H), \tau \Big) \ ,$$

Indistinguishability follows from the security of iO.

Hybrid 8**:** This hybrid is identical to **Hybrid** 1, with $\boldsymbol{x}'_H$ instead of $\boldsymbol{x}^*_H$. Indistinguishability from **Hybrid** 7 is identical to the indistinguishability of **Hybrid** 1 and **Hybrid** 7.

This completes the proof of Theorem 4.3. □

References

[ABT21] Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Perfect secure computation in two rounds. *SIAM J. Comput.*, 50(1):68–97, 2021.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in nc^0. *SIAM J. Comput.*, 36(4):845–888, 2006.

[AJLA$^+$12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 483–501, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, STOC '01, page 601–610, New York, NY, USA, 2001. Association for Computing Machinery.

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, pages 1–18, 2001.

[BGI$^+$14] Amos Beimel, Ariel Gabizon, Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, and Anat Paskin-Cherniavsky. Non-interactive secure multiparty computation. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 387–404. Springer, 2014.

[BHHO08] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision diffie-hellman. In David A. Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 108–125. Springer, 2008.

[BLP$^+$13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, page 575–584, New York, NY, USA, 2013. Association for Computing Machinery.

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, page 1–10, New York, NY, USA, 1988. Association for Computing Machinery.

[BP15] Nir Bitansky and Omer Paneth. Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 401–427. Springer, 2015.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 11–19. ACM, 1988.

[GGH$^+$13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[GM11] Vipul Goyal and Hemanta K. Maji. Stateless cryptographic protocols. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 678–687. IEEE Computer Society, 2011.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, page 218–229, New York, NY, USA, 1987. Association for Computing Machinery.

[GS09] Vipul Goyal and Amit Sahai. Resettably secure computation. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 54–71. Springer, 2009.

[HIJ+17] Shai Halevi, Yuval Ishai, Abhishek Jain, Ilan Komargodski, Amit Sahai, and Eylon Yogev. Non-interactive multiparty computation without correlated randomness. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part III*, volume 10626 of *Lecture Notes in Computer Science*, pages 181–211. Springer, 2017.

[HIKR23] Shai Halevi, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. Additive randomized encodings and their applications. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part I*, volume 14081 of *Lecture Notes in Computer Science*, pages 203–235. Springer, 2023.

[IKOS06] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography from anonymity. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 239–248. IEEE Computer Society, 2006.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 60–73. ACM, 2021.

[MV10] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC '10, page 351–358, New York, NY, USA, 2010. Association for Computing Machinery.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 427–437. ACM, 1990.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 333–342, New York, NY, USA, 2009. Association for Computing Machinery.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), sep 2009.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 543–553. IEEE Computer Society, 1999.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167, 1986.