# Additive Randomized Encodings from Public Key Encryption

Nir Bitansky[1,2], Saroja Erabelli[1], and Rachit Garg[1]

[1]New York University
[2]Tel Aviv University
nbitansky@gmail.com, saroja.erabelli@gmail.com,
rg5134@cims.nyu.edu

**Abstract**

Introduced by Halevi, Ishai, Kushilevitz, and Rabin (CRYPTO 2023), *Additive randomized encodings* (ARE) 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. The appeal of ARE comes from the simplicity of the non-local computation, involving only addition. This gives rise for instance to non-interactive secure function evaluation in the *shuffle model* where messages from different parties are anonymously shuffled before reaching their destination. Halevi, Ishai, Kushilevitz, and Rabin constructed ARE based on Diffie-Hellman type assumptions in bilinear groups.

We construct ARE assuming public-key encryption. The key insight behind our construction is that *one-sided ARE*, which only guarantees privacy for one of the parties, are relatively easy to construct, and yet can be lifted to full-fledged ARE. We also give a more efficient black-box construction from the CDH assumption.

## 1 Introduction

*Additive randomized encodings* (ARE), introduced by Halevi, Ishai, Kushilevitz, and Rabin [HIKR23], reduce the task of securely computing complex multi-party functions to simply computing addition over an Abelian group. Specifically, in an ARE for a $k$-party function $f(x_1, \ldots, x_k)$, each party $i$ locally computes a randomized encoding $\hat{x}_i$ of their input $x_i$ over an Abelian group $\mathbb{G}$. These encodings are then added together in $\mathbb{G}$ to yield an encoding $\hat{z} = \hat{x}_1 + \cdots + \hat{x}_k$ of the result $z = f(x_1, \ldots, x_k)$. A server, given the encoding $\hat{z}$ can decode the result $z$, but learns no additional information.

Such ARE can be seen as a variant of multi-party randomized encodings [AIK06, ABT21] where the simplicity of the joint encoding function is taken to an extreme. In the *shuffle model* [IKOS06], where parties can simultaneously send (multiple) anonymous messages to the server, ARE protocols give rise to *non-interactive* secure function computation [HIKR23]. This makes them especially appealing as a *setupless* alternative to other models of non-interactive secure computation that do require trusted setup such as correlated randomness or public-key infrastructure (c.f. [FKN94, BGI+14, HIJ+17, AAP19]).

So far the only existing constructions of ARE for general functions rely on either the Decision Squaring XDH Assumption in bilinear groups [HIKR23], or on Indistinguishability Obfuscation and DDH/LWE

[BF24].[1] Constructing ARE under weaker assumptions, as well as post-quantum assumptions, is posed as an open question in [HIKR23].

## 1.1 This Work

We show a generic construction of ARE based on any public-key encryption (PKE) scheme.

**Theorem 1.1** (Informal)**.** *Assuming public-key encryption, every efficient function has a computationally secure ARE.*

Public-key encryption schemes are as old as modern cryptography [DH76, RSA83, GM82], and are known by now based on a variety of number-theoretic assumptions, or code/lattice based assumptions that are conjectured to be post-quantum (see for instance Barak's survey [Bar17]).

**Main Conceptual Contribution: One-Sided ARE.** Our key idea is to consider a relaxed notion of *one-sided ARE*. Here for a two-party function $f(x, y)$, an ARE for $f$ may leak $x$, but hides $y$ (formally, the ARE can be simulated from $x, f(x, y)$). Assuming one-way functions, we show that any $f$ has such a one-sided ARE (alternatively, we can get an information-theoretic security with an inefficient ARE or if $f \in NC_1$).

Assuming public-key encryption, we then show how to lift any one-sided ARE to (two-sided) ARE, which yields our main result.

**A Simpler Construction Under CDH.** Our lifting transformation from one-sided ARE to ARE uses the underlying PKE protocol *in a non-black-box way*. We also give a simpler construction based on the CDH assumption (without any non-black-box use of cryptographic primitives). This is still an improvement compared to Square XDH in [HIKR23], and may be more practical than our main construction.

See further details in the technical overview below.

## 1.2 More Related Work

There is a long line of work concerning *randomized encodings* [IK00, AIK06] where the theme is to replace a possibly complex function $f$ with a *simpler* randomized encoding $\hat{f}$ of $f$ (see surveys [Ish13, App17]). ARE can be viewed as a randomized encoding, where the encoding is as simple as adding local functions.

ARE can also be viewed as variant of *multi-party randomized encodings* (MPRE) [ABT21]. In such encodings the goal is to simplify the *global* encoding function that aggregates the local encodings of parties' inputs. In the case of ARE, the global encoding is just addition, however, the security guarantee is inherently weaker than that of a MPRE. Indeed, MPRE may withstand corruption of parties whereas ARE only guarantees security against an external server.

The work of [HIKR23] also introduces the notion of *robust* ARE which offers security against corruptions as well. However, unlike MPRE where only the output is leaked, in robust ARE a set $C$ of corrupted parties may (inherently) learn the *residual function* $f(x_H, \cdot)$ where the honest parties' $H$ inputs are fixed. Such robust ARE turn out to be equivalent to obfuscation. In [HIKR23] a heuristic construction of simulation-based robust ARE is shown based on ideal obfuscation. Subsequently, in [BF24] an indistinguishability-based robust-ARE is constructed based on indistinguishability obfuscation and either DDH or LWE.

[1] The latter construction in fact achieves a stronger notion of robust ARE. See the related work subsection.

## 1.3 Technical Overview

We now explain the main ideas in our work. As mentioned above, the main enabler of our construction is the notion of *one-sided ARE* (OSARE). Our construction roughly consists of three main steps: (1) An information-theoretic construction of OSARE for the equality function. (2) A reduction of OSARE for general functions to OSARE for equality. (3) A lifting theorem from OSARE to full-fledged ARE.

The first step turns out to be an easy exercise, and the second step is a rather direct adaptation of a similar reduction for (two-sided) ARE in [HIKR23]. Our main technical contribution, beyond identifying OSARE as a potent notion, is in the third step of lifting OSARE to ARE. Accordingly, we start by overviewing this step and then give a brief overview of the first two steps.

Throughout, for two-party schemes, we refer by default to the first party as Alice and to the second party as Bob, and to their inputs as $x$ and $y$, respectively.

**One-Sided ARE.** Recall that in OSARE, we relax the security definition of ARE so that the server given the sum $\hat{x}+\hat{y}$, may learn not only the result $f(x,y)$ but also Alice's input $x$. Formally $\hat{x}+\hat{y}$ can be simulated from $(x, f(x,y))$.

**Naïve Idea: Encrypt Alice's $x$ and use SFE.** Aiming to leverage OSARE to construct a (two-sided) ARE for $f(x,y)$, we of course cannot use Alice's secret input $x$ directly (as it leaks). Instead, we'd like to use an encryption $\mathsf{Enc}_{\mathsf{k}}(x)$ and perform the computation over this encryption. For this purpose, we use perhaps the first primitive that comes to mind — *secure function evaluation* (SFE).

In an SFE scheme, the receiver Alice can send an encryption $\mathsf{rct} = \mathsf{Enc}_{\mathsf{k}}(x)$ of her input to the sender Bob, with respect to her secret key $\mathsf{k}$. Sender Bob then uses his input $y$ to generate an evaluated ciphertext $\mathsf{ect} = \mathsf{Eval}(\mathsf{rct}, y)$. Alice can then compute $\mathsf{Dec}_{\mathsf{k}}(\mathsf{ect})$ to decrypt the output $f(x,y)$. In terms of security, Alice is guaranteed that Bob learns nothing about her input $x$, and Bob is guaranteed that (a semi-honest) Alice learns nothing about his input $y$, except for $f(x,y)$. An external observer seeing only $(\mathsf{rct}, \mathsf{ect})$ does not learn anything at all.

Using SFE, the most straightforward idea is to use OSARE for the function

$$e(\mathsf{rct}, (y, r)) = \mathsf{Eval}(\mathsf{rct}, y; r) \ ,$$

with Alice's input set to $\mathsf{rct} = \mathsf{Enc}_{\mathsf{k}}(x)$ and Bob's input set to $y$ and the randomness $r$ needed for $\mathsf{Eval}$. OSARE security guarantees that the resulting (sum) encoding can be simulated from Alice's input $\mathsf{rct}$ and the result $\mathsf{ect}$, which by SFE security leaks nothing to the server.

However, this naïve attempt obviously falls short of obtaining our final goal, as the server also cannot learn $f(x,y)$, but only an encryption $\mathsf{ect}$ thereof.

**Controlled SFE Decryption using FE.** To complete the above idea, we aim to allow the server to decrypt $\mathsf{ect}$ in order to learn the result $f(x,y)$, but prevent it from decrypting $\mathsf{rct}$ and learning $x$. For this purpose, we use *functional encryption* (FE).

In a public-key FE scheme for a two-party function $d(\alpha,\beta)$, it is possible to generate a public encryption key $\mathsf{fpk}$ together with a master secret key $\mathsf{msk}$. Given $\mathsf{fpk}$ it is possible to generate an encryption $\mathsf{fct}_\alpha$ of any first input $\alpha$, and given $\mathsf{msk}$ it is possible to generate a function key $\mathsf{fsk}_\beta = \mathsf{KeyGen}_{\mathsf{msk}}(\beta)$ for any second input $\beta$. Given $(\mathsf{fct}_\alpha, \mathsf{fsk}_\beta)$ it is possible to decrypt the result $d(\alpha,\beta)$. The security guarantee is that nothing

is learned about $\alpha$ beyond $d(\alpha, \beta)$ (whereas $\beta$ may leak).[2] Such FE can be constructed based on plain public-key encryption [SS10]. (Importantly, we only require security in the presence of a single function key and do not pose any compactness requirements.)

Going back to our goal of allowing the server SFE decryption of ect, but nothing more, we consider an FE scheme for the SFE decryption function

$$d(\mathsf{k}, \mathsf{ect}) = \mathsf{SFE.Dec}_{\mathsf{k}}(\mathsf{ect}) \ .$$

We then augment our OSARE function $e(\mathsf{rct}, (y, r))$ from before to a new function

$$e(\mathsf{rct}, (y, r, \mathsf{msk})) = \mathsf{FE.KeyGen}_{\mathsf{msk}}(\mathsf{ect} = \mathsf{SFE.Eval}(\mathsf{rct}, y; r)) \ ,$$

which like before computes the SFE ciphertext ect, but then rather than outputting the ciphertext itself outputs a corresponding function key. Since the OSARE hides Bob's input $(y, r, \mathsf{msk})$, we are guaranteed the server only learns Alice's input rct and the resulting function key $\mathsf{fsk}_{\mathsf{ect}}$.

The only missing part is allowing the server to learn an FE encryption $\mathsf{FE.Enc}_{\mathsf{fpk}}(\mathsf{k})$ of the SFE secret key k. This would allow the server to obtain the decryption

$$f(x, y) = d(\mathsf{k}, \mathsf{ect}) = \mathsf{SFE.Dec}_{\mathsf{k}}(\mathsf{ect})$$

of the SFE ciphertext ect and nothing more.

For this, we simply use OSARE again, only this time in reverse. That is, the parties run an OSARE for the function

$$g(\mathsf{fpk}, (\mathsf{k}, r')) = \mathsf{FE.Enc}_{\mathsf{fpk}}(\mathsf{k}; r')$$

that encrypts k under the FE public key fpk using randomness $r'$. This time around, Alice provides the secret input $(\mathsf{k}, r')$ whereas Bob provides the leaky input fpk. The OSARE guarantees that the server only learns the result $\mathsf{fct} = \mathsf{FE.Enc}_{\mathsf{fpk}}(\mathsf{k}; r')$ and the leaky input fpk.

Overall, considering the two instances of OSARE, the server only learns

$$\mathsf{rct}, \mathsf{fsk}_{\mathsf{ect}}, \mathsf{fpk}, \mathsf{fct} \ ,$$

which by the security of FE and SFE reveal nothing but $f(x, y)$.

**PKE instead of SFE.** SFE protocols are known to be equivalent to two-message (semi-honest) oblivious transfer [Yao86]. We observe, however, that traditional SFE gives more than we bargained for. Indeed, SFE guarantees security when one of the parties is corrupted, whereas in our setting we only need security against an external observer that only sees the ciphertexts rct, ect, but is completely oblivious to the inner state of either party. In other words, we only need secure communication, which we can achieve using (plain) public-key encryption. In the language we have used so far, we can think of an SFE protocol, where Alice sends her public encryption key pk and Bob sends back an encryption $\mathsf{Enc}_{\mathsf{pk}}(y)$, Alice then decrypts, and computes $f(x, y)$ herself.

Let us redescribe the scheme, forgoing SFE altogether. In the augmented scheme, instead of generating an SFE key, Alice generates PKE keys $(\mathsf{pk}, \mathsf{sk})$. For our FE scheme, we replace the SFE decryption function with the PKE decryption function along with the function evaluation of $f$:

$$d((x, \mathsf{sk}), \mathsf{pct}) = f(x, \mathsf{PKE.Dec}_{\mathsf{sk}}(\mathsf{pct})) \ .$$

[2] Typically, the fixed function $d$ is thought of as a universal circuit, and the second input $b$ describes a circuit for a dynamically chosen function. The above equivalent formulation is more convenient in our setting.

We modify the OSARE function to output a corresponding function key:

$$e(\mathsf{pct}, (y, r, \mathsf{msk})) = \mathsf{FE.KeyGen}_{\mathsf{msk}}(\mathsf{pct} = \mathsf{PKE.Enc}(y; r)) \ .$$

The second OSARE now encrypts the secret key sk and Alice's input $x$ together, instead of the SFE secret key k:

$$g(\mathsf{fpk}, (x, \mathsf{sk}, r')) = \mathsf{FE.Enc}_{\mathsf{fpk}}((x, \mathsf{sk}); r') \ ,$$

where Alice provides the secret input $(x, \mathsf{sk}, r')$ and Bob provides the leaky input fpk. In the new scheme, the server only learns

$$\mathsf{pct}, \mathsf{fsk}_{\mathsf{pct}}, \mathsf{fpk}, \mathsf{fct} \ ,$$

which reveals nothing but $f(x, y)$ by the security of FE and PKE. Finally, we note that two-party (full-fledged) ARE (even just for equality) imply multi-party (full-fledged) ARE, by a reduction from [HIKR23] (we in fact review this reduction below, observing that it also applies to OSARE).

**From OSARE for Equality to OSARE for General Functions.** We review the reduction of ARE for general functions to ARE for the equality function [HIKR23], and observe that the exact same reduction also applies for OSARE. Specifically, consider the two-party function that checks equality of two elements $x, y$ from a finite domain $[d]$.

We first show that this implies an OSARE for any function $f : [d-1] \times [d-1] \to \{0, 1\}$. To achieve this, Alice and Bob execute $d-1$ OSARE protocols for equality. Alice uses her input $x$ in all of them. Bob lists all inputs $x'$ such that $f(x', y) = 1$, pads the list to length $d-1$, using the element $d$, which is outside the function domain, and randomly permutes (or just shifts) the list. The OSARE only reveals Alice's input $x$, and each of the equality results, which are either all non-equal when $f(x, y) = 0$, or equal in a single random instance in case $f(x, y) = 1$.

This in particular implies an OSARE for the OT function $(c, (s_0, s_1)) \mapsto s_c$, where the server only learn a chosen string $s_c$, as well as the choice bit $c$. Then, we combine the OSARE for OT with Yao's garbled circuit to get an OSARE for general two-party functions $f(x, y)$. Here Bob would generate the garbled circuit corresponding to $f(\cdot, y)$ and Alice would OT-choose the labels corresponding to her input $x$. OSARE for OT guarantees that the server learns only Alice's input $x$ as well as the garbled circuits and corresponding $x$-labels, which only reveal $f(x, y)$.

**A Statistical OSARE for Equality.** To construct a one-sided ARE for equality for input domain $D = [d]$, we encode elements over $\mathbb{Z}_t^d$ (for large $t$). On input $x$, Alice outputs a length $d$ vector with 0 at the $x^{th}$ entry and random elements in all other entries. On input $y$, Bob does the complement: outputting a length $d$ vector with a random element at the $y^{th}$ entry and 0 in all other entries. The decoder outputs `equal` if all entries are nonzero and `not-equal` otherwise. If $x = y$, the scheme is perfectly simulatable (it in fact does not leak anything). If $x \neq y$, the scheme leaks $x$, but nothing on $y$, except that $y \neq x$. As described the scheme has perfect security and correctness error $O(1/t)$. Alternatively, by restricting to non-zero elements, we can get perfect correctness and statistical security error $O(1/t)$.

We note that in [HIKR23], an ARE for an "equality-type" function is given, which in fact can also be shown to be a OSARE for the (standard) equality function. However, there it is not clear how to obtain perfect correctness.

**ARE from CDH.** Finally, we briefly overview the direct construction of ARE from the CDH assumption.

Our first observation is that the pairing-based construction from [HIKR23] can easily be modified to use the Squaring DDH assumption in plain groups. In their construction, the additive encodings are of the form

$$(\hat{x}_0, \hat{x}_1, \hat{x}_2) = ([r]_1, [rx]_2, [rx^2]_t)$$
$$(\hat{y}_0, \hat{y}_1, \hat{y}_2) = ([s]_1, [-sy]_2, [-sy^2]_t)$$

where $[a]_1, [b]_2, [c]_t$ denote values $g_1^a, g_2^b, g_t^c$ in the pairing groups $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_t$ with generators $g_1, g_2$, and $g_t$, respectively. To check equality, the evaluator obtains the sum $(\hat{z}_0, \hat{z}_1, \hat{z}_2) = ([r+s]_1, [rx-sy]_2, [rx^2-sy^2]_t)$ and checks whether $e(\hat{z}_0, \hat{z}_1) = \hat{z}_2$, which always occurs when $x = y$, and with small probability otherwise. We observe that security, in fact, still holds even if one of the exponents is given in the clear:

$$(\hat{x}_0, \hat{x}_1, \hat{x}_2) = (r, [rx], [rx^2])$$
$$(\hat{y}_0, \hat{y}_1, \hat{y}_2) = (s, [-sy], [-sy^2])$$

where $[a]$ denotes $g^a$ in the DDH group $\mathbb{G}$, and the equality check remains the same as in the first construction, but now does not require pairing. We then show how to apply the Goldreich Levin hard-core bit theorem to further reduce the construction to the Squaring CDH assumption, which is in turn equivalent to plain CDH [MW99, BDZ03]. More details are provided in section 4.

# 2 Preliminaries

Throughout this work, we denote by $\lambda$ the security parameter. We say a function $f$ is negligible in the security parameter $\lambda$ if $f = \lambda^{-\omega(1)}$. We denote this by writing $f(\lambda) = \mathrm{negl}(\lambda)$. We write $\mathrm{poly}(\lambda)$ to denote a function that is bounded by a fixed polynomial in $\lambda$. We say an algorithm is efficient if it runs in probabilistic polynomial time (PPT) in the length of its input.

We use $\mathbb{Z}_t^\times$ to denote $\mathbb{Z}_t \setminus \{0\}$. For a positive integer $n$, we use $[n]$ to denote the set $\{1, \ldots, n\}$ . For a set $S$, $x \leftarrow S$ denotes uniformly sampling from $S$. We use $\mathcal{U}_n$ to denote the uniform distribution over $\{0,1\}^n$. We next review the main cryptographic primitives we use in this work.

## 2.1 Additive Randomized Encodings

**Definition 2.1** (Additive Randomized Encoding [HIKR23])**.** Let $k(\lambda), n(\lambda), m(\lambda)$ be polynomially bounded functions. A $k$-party ARE scheme for a function $f : (\{0,1\}^n)^k \to \{0,1\}^m$, consists of three PPT algorithms $\Pi = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax:

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

We require the following properties:

- Correctness: $\Pi$ is $\varepsilon$-correct, for a function $\varepsilon(\lambda)$, if for all $\lambda$ and $x_1 \ldots, x_k \in \{0,1\}^n$:

$$\Pr\left[\mathsf{Dec}_{\mathsf{pp}}(\widehat{y}) = f(x_1, \ldots, x_k) : \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ \widehat{x}_i \leftarrow \mathsf{Enc}_{\mathsf{pp}}(x_i, i) \\ \widehat{y} = \sum_{i=1}^{k} \widehat{x}_i \end{array}\right] \geq 1 - \varepsilon(\lambda) \ .$$

- Security: There exists a PPT simulator Sim such that for any $\lambda$, and $x_1, \ldots, x_k \in \{0,1\}^n$,:

$$(\mathsf{pp}, \widehat{y}) \approx_c \mathsf{Sim}(1^\lambda, f(x_1, \ldots, x_k)) \ ,$$

  where $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $\widehat{x}_i \leftarrow \mathsf{Enc}_{\mathsf{pp}}(x_i, i)$, and $\widehat{y} = \sum_{i=1}^{k} \widehat{x}_i$.

  We further say that security is *statistical/perfect* if instead of computational indistinguishability we have statistical/perfect indistinguishability.

## 2.2 Public Key Encryption

**Definition 2.2** (Public Key Encryption)**.** Let $n(\lambda)$ be a polynomially bounded function. A public key encryption scheme consists of three algorithms $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$ given the security parameter $\lambda$, outputs a public/secret key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{pct} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(x)$ is a randomized algorithm, given input $x \in \{0,1\}^n$, public key $\mathsf{pk} \in \{0,1\}^\lambda$, outputs a ciphertext pct.
- $z \leftarrow \mathsf{Dec}_{\mathsf{sk}}(\mathsf{pct})$ is a deterministic algorithm that given a ciphertext pct and secret key $\mathsf{sk} \in \{0,1\}^\lambda$, outputs a result $z$.

We require the following properties:

- Correctness: For all $\lambda \in \mathbb{N}$ and $x \in \{0,1\}^n$,

$$\Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathsf{pct}) = x] = 1 \ .$$

  where $\mathsf{pct} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(x)$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$.

- Semantic Security: For all $\lambda \in \mathbb{N}$, and all inputs $x \in \{0,1\}^n$,

$$\left(\mathsf{pk}, \mathsf{Enc}_{\mathsf{pk}}(x)\right) \approx_c \left(\mathsf{pk}, \mathsf{Enc}_{\mathsf{pk}}(0^n)\right) \ ,$$

  where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$.

## 2.3 Single-Key Functional Encryption

**Definition 2.3** (Single-Key Functional Encryption)**.** Let $n_X(\lambda), n_Y(\lambda), m(\lambda)$ be polynomially bounded functions. A single-key functional encryption scheme for $f : \{0,1\}^{n_X} \times \{0,1\}^{n_Y} \rightarrow \{0,1\}^m$ consists of four algorithms $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax:

- $(\mathsf{msk}, \mathsf{fpk}) \leftarrow \mathsf{Setup}(1^\lambda)$ given the security parameter $\lambda$, outputs a master secret key msk and a public key fpk.

- $\mathsf{fsk}_y \leftarrow \mathsf{KeyGen}(\mathsf{msk}, y)$ given the master secret key msk and $y \in \{0,1\}^{n_Y}$, outputs a functional secret key $\mathsf{fsk}_y$ corresponding to the function $f(\cdot, y)$.
- $\mathsf{fct} \leftarrow \mathsf{Enc}_{\mathsf{fpk}}(x)$ given $x \in \{0,1\}^{n_X}$ and public key fpk, generates a ciphertext fct.
- $z \leftarrow \mathsf{Dec}_{\mathsf{fsk}_y}(\mathsf{fct})$ given a ciphertext fct and function key $\mathsf{fsk}_y$ outputs a decryption value $z$.

We require the following properties:

- Correctness: For all $\lambda \in \mathbb{N}$ and $x, y \in \{0,1\}^{n_X} \times \{0,1\}^{n_Y}$,

$$\mathsf{Dec}_{\mathsf{fsk}_y}(\mathsf{fct}) = f(x, y) \ .$$

  where $\mathsf{fct} \leftarrow \mathsf{Enc}_{\mathsf{fpk}}(x)$ and $\mathsf{fsk}_y = \mathsf{KeyGen}(\mathsf{msk}, y)$.

- Security: There exists a PPT algorithm Sim such that for all $\lambda \in \mathbb{N}$, and all inputs $x, y \in \{0,1\}^{n_X} \times \{0,1\}^{n_Y}$,

$$(\mathsf{fpk}, \mathsf{fsk}_y, \mathsf{Enc}_{\mathsf{fpk}}(x)) \approx_c \mathsf{Sim}(y, f(x, y)) \ ,$$

  where the probability distributions are taken over the randomness of FE.Sim and Enc.

**Remark 2.4.** Commonly in the literature, one considers single-input functions $f(x)$ of some bounded circuit size and functional keys $\mathsf{fsk}_f$. The above formulation is equivalent and will be more convenient for our purposes.

**Instantiation.** FE schemes as above are known for any polynomially computable $f$ based on plain public-key encryption [SS10].

# 3 ARE from PKE and FE

## 3.1 One-Sided ARE

We define a weaker notion of an ARE, which we call a one-sided ARE below, where security only guarantees the privacy of one party's input.

**Definition 3.1** (One-Sided Additive Randomized Encoding)**.** An ARE scheme $\Pi = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ for $f : \{0,1\}^{n_X} \times \{0,1\}^{n_Y} \to \{0,1\}^m$ defined for two parties is said to be a one-sided ARE if it satisfies the same correctness guarantee as an ARE and the security guarantee below:

One-Sided Security: There exists a PPT simulator Sim such that for any $\lambda$, and $x \in \{0,1\}^{n_X}, y \in \{0,1\}^{n_Y}$:

$$(\mathsf{pp}, \widehat{z}) \approx_c \mathsf{Sim}(1^\lambda, x, f(x, y)) \ ,$$

where $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $\widehat{x} \leftarrow \mathsf{Enc}_{\mathsf{pp}}(x, 1)$, $\widehat{y} \leftarrow \mathsf{Enc}_{\mathsf{pp}}(y, 2)$, and $\widehat{z} = \widehat{x} + \widehat{y}$.

We further say that security is *statistical/perfect* if instead of computational indistinguishability we have statistical/perfect indistinguishability.

**One-Sided ARE Scheme $\mathsf{ARE}_{eq}$ for Equality**

$\underline{\mathsf{Enc}(1^\lambda, i, 1)}$:

- Sample $u_1, \ldots, u_d \leftarrow \mathbb{Z}_t^\times$.
- Set $\widehat{i}$ to be the $d$-dimensional vector $\sum_{k \neq i} u_k \cdot \boldsymbol{e}_k$, where $\boldsymbol{e}_k$ is the $k$-th standard basis vector.

$\underline{\mathsf{Enc}(1^\lambda, j, 2)}$:

- Sample $v \leftarrow \mathbb{Z}_t^\times$.
- Set $\widehat{j}$ to be the $d$-dimensional vector $v \cdot \boldsymbol{e}_j$.

$\underline{\mathsf{Dec}(1^\lambda, \widehat{w}) :}$

- Parse the (sum) encoding $\widehat{w}$ as a vector $(\widehat{w}_1, \ldots, \widehat{w}_d)$.
- If for all $k$, $\widehat{w}_k \neq 0$, output `equal`, else output `not-equal`.

Figure 1: One-Sided ARE for Equality

## 3.2 One-Sided ARE for Equality

We describe a one-sided ARE $\mathsf{ARE}_{eq}$ for the equality function over a finite domain $D = \{1, \ldots, d\}$. In the constructions below, we do not require a public setup, and omit the setup when not needed. Instead we pass the security parameter explicitly to the encoding and decoding algorithms. Let $t(\lambda)$ be a function, where $t \approx 2^\lambda$ by default.

**Proposition 3.2.** *The scheme $\mathsf{ARE}_{eq}$ is perfectly correct and $O(t^{-1})$-statistically secure.*

*Proof.* We prove that the scheme satisfies the two properties required by a one-sided ARE.

**Correctness:**
Our encodings are computed additively, i.e. $\widehat{w} = \widehat{i} + \widehat{j}$.

- When inputs are equal, i.e. if $i = j = k$.

  For any index $\ell \neq k \in [d]$, observe that $\widehat{w}_\ell = u_\ell \neq 0$ and for index $\ell = k$, $\widehat{w}_\ell = v \neq 0$. Since all coordinates are non-zero, decryption outputs `equal`.

- When inputs are unequal, i.e. if $i \neq j$.

  In this case, $\widehat{w}_i = 0$ and decryption outputs `not-equal`.

**One-Sided Security:**

- When inputs are equal, $\mathsf{Sim}(1^\lambda, \mathtt{equal})$ randomly samples $(\widehat{w}_1, \ldots, \widehat{w}_d) \leftarrow (\mathbb{Z}_t^\times)^d$, and perfectly emulates the encoding $\widehat{i} + \widehat{j} = \sum_{l \neq i} u_l \cdot \boldsymbol{e}_l + v \cdot \boldsymbol{e}_j$.

- When inputs are unequal, $\mathsf{Sim}(1^\lambda, i, \texttt{not-equal})$ sets $\widehat{w}_i = 0$, for $\ell \neq i$ samples $\widehat{w}_\ell \leftarrow \mathbb{Z}_t^\times$, and outputs $(\widehat{w}_1, \ldots, \widehat{w}_d)$.

  This distribution perfectly emulates all coordinates $\ell \neq j$ of the encoding $\widehat{i} + \widehat{j} = \sum_{\ell \notin \{i,j\}} u_\ell \cdot \boldsymbol{e}_\ell + (v + u_j) \cdot \boldsymbol{e}_j$. For the coordinate $j$, the statistical distance between $v + u_j \sim U(\mathbb{Z}_t^\times) + U(\mathbb{Z}_t^\times)$ and $\widehat{w}_j \sim U(\mathbb{Z}_t^\times)$ is $O(t^{-1})$.

□

### 3.3 From Equality to Any Small Function

We observe that the transformation of [HIKR23] from ARE for Equality to ARE for any finite domain applies also for the case of one-sides AREs. That is, one-sided ARE for Equality implies one-sided ARE for any finite-domain function. Below, the proof is taken almost verbatim from [HIKR23], adopted to the case of one-sided security.

**Lemma 3.3** (Adaptation of [HIKR23, Lemma 5.7])**.** *Let $f : D_1 \times D_2 \to \{0,1\}$ be a Boolean function over finite domains $D_1$ and $D_2$. Assume w.l.o.g. that $|D_1| \leq |D_2|$, and let $z$ be an arbitrary symbol $z \notin D_1$. Then a secure one-sided ARE scheme $\mathsf{OSARE}_{eq}$ for equality over the domain $D = D_1 \cup \{z\}$ can be converted into a secure one-sided ARE scheme $\mathsf{OSARE}_f$ for function $f$, where the communication complexity of $\mathsf{OSARE}_f$ is at most $|D_1|$ times larger than that of $\mathsf{OSARE}_{eq}$.*

*Proof.*

**The scheme $\mathsf{OSARE}_f$.** Let $d_1 = |D_1|$. On inputs $x \in D_1, y \in D_2$, the parties run $d_1$ parallel copies of the equality scheme $\mathsf{OSARE}_{eq}$:

- Party 1 uses their input $x$ in all these copies.
- For Party 2, let $x_1, x_2, \ldots, x_d$ be all the possible party-1 inputs for which $f(x_i, y) = 1$. Party 2 concatenates $d_1 - d$ copies of the values $z \notin D_1$, yielding $a = (x_1, \ldots, x_d, z, \ldots, z) \in D^{d_1}$. Party 2 then shifts by a randomly chosen $\delta \leftarrow [0, d_1 - 1]$ to obtain $b$ such that the $i^{th}$ entry of $b$ is the $(i - \delta) \pmod{d_1}$ entry of $a$.

The evaluator gets $d_1$ sums $y_1, \ldots, y_{d_1}$ from the $d_1$ copies of $\mathsf{OSARE}_{eq}$ and decodes them to get $d_1$ results $b_i = \mathsf{Dec}_{eq}(y_i)$. It outputs 1 if there is any match $b_i = \texttt{equal}$, and outputs 0 if they are all $\texttt{not-equal}$.

We prove that $\mathsf{OSARE}_f$ satisfies the two properties required by a one-sided ARE.

**Correctness.** Since Party 1 uses $x \in D_1$ in all the copies of $\mathsf{OSARE}_{eq}$ and Party 2 uses $d$ distinct inputs from $D_1$ and the value $z \notin D_1$, then at most one of them will be a match. A match occurs if and only if $x = x_i$ for some $i \leq d$, which means that $f(x, y) = f(x_i, y) = 1$.

**One-Sided Security.** Let $\mathsf{Sim}_{eq}$ be the one-sided security simulator for the scheme $\mathsf{ARE}_{eq}$.

- When $f(x, y) = 0$, $\mathsf{Sim}_f(1^\lambda, x, 0)$ computes $\forall i \in [d_1]$, $v_i \leftarrow \mathsf{Sim}_{eq}(1^\lambda, x, \texttt{not-equal})$, and outputs $(v_1, \ldots, v_k)$. This statistically emulates the encoding by a standard hybrid argument.

- When $f(x,y) = 1$, $\mathsf{Sim}_f(1^\lambda, x, 1)$ samples $i \leftarrow [d_1]$, computes $v_i \leftarrow \mathsf{Sim}_{eq}(1^\lambda, x, \texttt{equal})$, and $\forall j \in [d_1],\ j \neq i,\ v_j \leftarrow \mathsf{Sim}_{eq}(1^\lambda, x, \texttt{not-equal})$, and outputs $(v_1, \ldots, v_k)$. Similarly, this statistically emulates the encoding by a standard hybrid argument.

□

### 3.3.1 One-Sided ARE for General Functions

In addition, the transformation of [HIKR23] from ARE for Equality to ARE for general functions applies also for the case of one-sides AREs. That is, one-sided ARE for Equality implies one-sided ARE for general functions. Below, the proof is also taken almost verbatim from [HIKR23], adopted to the case of one-sided security.

**Lemma 3.4** (Adaptation of [HIKR23, Lemma 5.8])**.** *Let $n_X(\lambda), n_Y(\lambda), m(\lambda)$ be polynomially bounded functions. Let $f : \{0,1\}^{n_X} \times \{0,1\}^{n_Y} \to \{0,1\}^m$ be a two-party function, and let $\{C_\lambda\}$ be a family of polynomial-size circuits computing $f$. Assuming one-way functions, there exists a computationally-secure one-sided ARE scheme $\mathsf{OSARE}_f$ for $f$. Furthermore, we obtain statistical one-sided ARE (without assuming one-way functions) if we allow an inefficient ARE scheme, or if $\{C^\lambda\} \in \mathsf{NC}^1$.*

*Proof Sketch.* Let $\mathsf{OSARE}_{sot}$ be a one-sided ARE scheme, which we obtain from Lemma 3.3, for the string oblivious transfer function:

$$f_{sot} : \{0,1\} \times \{0,1\}^{2\lambda} \to \{0,1\}^\lambda, f_{sot}(c, (s_0, s_1)) = s_c \ .$$

We build a one-sided ARE scheme $\mathsf{OSARE}_f$ for $f$. If we allow an inefficient ARE scheme, then we can simply apply Lemma 3.3 to construct $\mathsf{OSARE}_f$. Otherwise, party 2 will construct and send the evaluator a garbling of $C_\lambda$ (which can be implemented from any one-way function or by using information-theoretic garbling [IK02] if $\{C^\lambda\} \in \mathsf{NC}^1$ and will run with Party 1 $n_X$ instances of $\mathsf{OSARE}_{sot}$ for strings of length $\lambda$, one for each of their input bits. In each instance, Party 2 will play the role of the sender using as input the two labels for that input wire, and Party 1 will play the receiver using the corresponding input bit as the OT choice bit. The evaluator will therefore receive the garbled circuit, along with one label for each input wire. It will then evaluate the garbled circuit and compute the output. Correctness and security follow from those of $\mathsf{OSARE}_{sot}$ and the garbling scheme. Unlike [HIKR23], the security of $\mathsf{OSARE}_{sot}$ is only one-sided, so simulating the encoding requires knowing $x$. Accordingly, simulating the encoding for $f$ requires knowing $x$, so accordingly we only obtain one-sided security.

## 3.4 One-Sided ARE to Full-Fledged ARE from PKE

In this section, we show how to transform one-sided ARE for general functions to (two-sided) ARE for general functions, assuming PKE and FE, which in turn can be constructed from PKE. Note that two-party full-fledged ARE can be transformed into $k$-party ARE via [HIKR23, Lemma 5.8].

**Construction 3.5** (ARE from One-Sided ARE using PKE)**.** Throughout, we let $\lambda$ be the security parameter. Let $f(x,y)$ be a function that we wish to compute using our ARE, where $x$ is the input of the first party and $y$ is the input of the second party.

- Let $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme (definition 2.2).

- Let $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a single-key functional encryption scheme (definition 2.3) for the evaluation function
$$d((x, \mathsf{sk}), \mathsf{pct}) := f(x, \mathsf{PKE.Dec}_{\mathsf{sk}}(\mathsf{pct})) \ ,$$
which given PKE secret key sk and PKE ciphertext pct, decrypts the ciphertext and evaluates $f$.
- Let $\mathsf{OSARE}_g = (\mathsf{Enc}, \mathsf{Dec})$ be a one-sided ARE scheme (definition 3.1) for the key-generation function,
$$g(\mathsf{pk}, (\mathsf{msk}, y, r_{\mathsf{PKE}})) := \mathsf{FE.KeyGen}(\mathsf{msk}, \mathsf{pct}),$$
$$\text{where } \mathsf{pct} := \mathsf{PKE.Enc}_{\mathsf{pk}}(y, r_{\mathsf{PKE}}) \ .$$
That is, the function $g$ given a PKE ciphertext pct and master secret key msk, input $y$, and PKE encryption randomness $r_{\mathsf{PKE}}$, encrypts $y$ and outputs a function secret key $\mathsf{fsk}_{\mathsf{pct}}$ corresponding to the function $d(\cdot, \mathsf{pct})$.
- Let $\mathsf{OSARE}_e$ be a one-sided ARE scheme (definition 3.1) for an encryption function,
$$e(\mathsf{fpk}, (x, \mathsf{sk}, r_{\mathsf{FE}})) := \mathsf{FE.Enc}_{\mathsf{fpk}}((x, \mathsf{sk}) ; r_{\mathsf{FE}}) \ ,$$
which given a public key, and message k, outputs an FE encryption.

**Remark 3.6.** Note that our full-fledged ARE scheme, as described, does not require setup since our one-sided ARE schemes also do not require setup. The only public parameter is $1^\lambda$.

**Proposition 3.7.** *The scheme* $\mathsf{ARE}_f$ *is perfectly correct.*

*Proof.* From correctness of the one-sided ARE schemes $\mathsf{OSARE}_g, \mathsf{OSARE}_e$, we have
$$\mathsf{fsk}_{\mathsf{pct}} = \mathsf{OSARE}_g.\mathsf{Dec}(\widehat{z}_0) = g(\mathsf{pk}, (\mathsf{msk}, y, r_{\mathsf{PKE}})) = \mathsf{FE.KeyGen}(\mathsf{msk}, \mathsf{pct}) \ ,$$
where $\mathsf{pct} = \mathsf{PKE.Enc}_{\mathsf{pk}}(y, r_{\mathsf{PKE}})$ and
$$\mathsf{fct} = \mathsf{OSARE}_e.\mathsf{Dec}(\widehat{z}_1) = e(\mathsf{fpk}, (x, \mathsf{sk}, r_{\mathsf{FE}})) = \mathsf{FE.Enc}_{\mathsf{fpk}}((x, \mathsf{sk}) ; r_{\mathsf{FE}}) \ ,$$
respectively. From correctness of the FE and PKE schemes, we have,
$$\mathsf{Dec}(1^\lambda, \widehat{z} = (\widehat{z}_0, \widehat{z}_1)) = \mathsf{FE.Dec}_{\mathsf{fsk}_{\mathsf{pct}}}(\mathsf{fct}) = d((x, \mathsf{sk}), \mathsf{pct}) := f(x, \mathsf{PKE.Dec}_{\mathsf{sk}}(\mathsf{pct})) = f(x, y) \ .$$
□

**Proposition 3.8.** *The scheme* $\mathsf{ARE}_f$ *is computationally secure.*

*Proof.* Simulator $\mathsf{ARE}_f.\mathsf{Sim}(1^\lambda, f(x, y))$ outputs $\widehat{z}' = (\widehat{z}'_0, \widehat{z}'_1)$ and is defined below.

Compute:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$.
- $\mathsf{pct}' \leftarrow \mathsf{PKE.Enc}_{\mathsf{pk}}(0^{n_Y})$.

**Full-Fledged Two-Party ARE** $\mathsf{ARE}_f$

$\underline{\mathsf{Enc}(1^\lambda, x, 1)}$:

- Sample $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$.
- Compute $\widehat{x}_0 \leftarrow \mathsf{OSARE}_g.\mathsf{Enc}(\mathsf{pk}, 1)$.
- Compute $\widehat{x}_1 \leftarrow \mathsf{OSARE}_e.\mathsf{Enc}((x, \mathsf{sk}), 2)$, (recall that the second party's inputs are hidden in this one-sided ARE scheme).
- Output $\widehat{x} = (\widehat{x}_0, \widehat{x}_1)$.

$\underline{\mathsf{Enc}(1^\lambda, y, 2)}$:

- Sample $r_{\mathsf{PKE}} \leftarrow \{0,1\}^\lambda$ and compute $(\mathsf{msk}, \mathsf{fpk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$.
- Compute $\widehat{y}_0 \leftarrow \mathsf{OSARE}_g.\mathsf{Enc}((\mathsf{msk}, y, r_{\mathsf{PKE}}), 2)$, (recall that the second party's inputs are hidden in this one-sided ARE scheme).
- Compute $\widehat{y}_1 \leftarrow \mathsf{OSARE}_e.\mathsf{Enc}(\mathsf{fpk}, 1)$.
- Output $\widehat{y} = (\widehat{y}_0, \widehat{y}_1)$ .

$\underline{\mathsf{Dec}\left(1^\lambda, \widehat{z} = (\widehat{z}_0, \widehat{z}_1)\right)}$:

- Compute $\mathsf{fsk}_{\mathsf{pct}} \leftarrow \mathsf{OSARE}_g.\mathsf{Dec}(1^\lambda, \widehat{z}_0)$.
- Compute $\mathsf{fct} \leftarrow \mathsf{OSARE}_e.\mathsf{Dec}(1^\lambda, \widehat{z}_1)$.
- Output $\mathsf{FE.Dec}_{\mathsf{fsk}_{\mathsf{ect}}}(\mathsf{fct})$ .

Figure 2: Full-Fledged Two-Party ARE from One-Sided ARE for a function $f$

- $(\mathsf{fpk}', \mathsf{fsk}', \mathsf{fct}') \leftarrow \mathsf{FE.Sim}(1^\lambda, \mathsf{pct}', f(x, y))$.
- $\widehat{z}'_0 \leftarrow \mathsf{OSARE}_g.\mathsf{Sim}(1^\lambda, \mathsf{pk}, \mathsf{fsk}')$.
- $\widehat{z}'_1 \leftarrow \mathsf{OSARE}_e.\mathsf{Sim}(1^\lambda, \mathsf{fpk}', \mathsf{fct}')$.

Output $\widehat{z}' = \left(\widehat{z}'_0, \widehat{z}'_1\right)$.

We define a sequence of hybrids starting with a hybrid that captures real encodings produced by $\mathsf{ARE}_f$ and ending with simulated encodings produced by $\mathsf{ARE}_f.\mathsf{Sim}$.

**Hybrid** 0**.** The algorithm presented in construction 3.5:

1. Compute $(\widehat{x}_0, \widehat{x}_1) \leftarrow \mathsf{Enc}(1^\lambda, x, 1)$.

(a) Sample $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$.

(b) Compute $\widehat{x}_0 \leftarrow \mathsf{OSARE}_g.\mathsf{Enc}(\mathsf{pk}, 1)$ and $\widehat{x}_1 \leftarrow \mathsf{OSARE}_e.\mathsf{Enc}((x, \mathsf{sk}), 2)$.

2. Compute $(\widehat{y}_0, \widehat{y}_1) \leftarrow \mathsf{Enc}(1^\lambda, y, 2)$.

(a) Sample $r_{\mathsf{PKE}} \leftarrow \{0,1\}^\lambda$ and compute $(\mathsf{msk}, \mathsf{fpk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$.

(b) Compute $\widehat{y}_0 \leftarrow \mathsf{OSARE}_g.\mathsf{Enc}((\mathsf{msk}, y, r_{\mathsf{PKE}}), 2)$ and $\widehat{y}_1 \leftarrow \mathsf{OSARE}_e.\mathsf{Enc}(\mathsf{fpk}, 1)$.

3. Output $(\widehat{z}_0 = \widehat{x}_0 + \widehat{y}_0, \widehat{z}_1 = \widehat{x}_1 + \widehat{y}_1)$.

**Hybrid** 1**.** We sample the encoding $\widehat{z}'_0$ using the simulator $\mathsf{OSARE}_g.\mathsf{Sim}$ instead of the real encoder which computes $\widehat{x}_0, \widehat{y}_0$ and sets $\widehat{z}_0 = \widehat{x}_0 + \widehat{y}_0$.

More concretely, we set $\widehat{z}'_0 \leftarrow \mathsf{OSARE}_g.\mathsf{Sim}(1^\lambda, \mathsf{pk}, \mathsf{fsk}_{\mathsf{pct}})$ where $\mathsf{pct} \leftarrow \mathsf{PKE.Enc}(y; r_{\mathsf{PKE}})$, and $\mathsf{fsk}_{\mathsf{pct}} \leftarrow \mathsf{FE.KeyGen}(\mathsf{msk}, \mathsf{pct})$.

**Hybrid** 2**.** We sample the encoding $\widehat{z}'_1$ using the simulator $\mathsf{OSARE}_e.\mathsf{Sim}$ instead of the real encoder which computes $\widehat{x}_1, \widehat{y}_1$ and sets $\widehat{z}_1 = \widehat{x}_1 + \widehat{y}_1$.

More concretely, we set $\widehat{z}'_1 \leftarrow \mathsf{OSARE}_e\mathsf{Sim}(1^\lambda, \mathsf{fpk}, \mathsf{fct})$ where $\mathsf{fct} \leftarrow \mathsf{FE.Enc}_{\mathsf{fpk}}((x, \mathsf{sk}); r_{\mathsf{FE}})$.

**Hybrid** 3**.** We simulate the functional secret key and ciphertext using the simulator $\mathsf{FE.Sim}$ instead of running $\mathsf{FE.Setup}$, $\mathsf{FE.Enc}$, $\mathsf{FE.KeyGen}$ algorithms.

More concretely, we set $(\mathsf{fpk}', \mathsf{fsk}', \mathsf{fct}') \leftarrow \mathsf{FE.Sim}(1^\lambda, \mathsf{pct}, f(x, y))$.

**Hybrid** 4**.** We simulate the public key encryption ciphertext by encrypting $0^{n_Y}$ instead of $y$.

More concretely, we set $\mathsf{pct}' \leftarrow \mathsf{PKE.Enc}_{\mathsf{pk}}(0^{n_Y})$.

We write $\mathsf{Hyb}_i$ to denote the output distribution of hybrid $i$. We now show that each pair of adjacent distributions defined above are computationally indistinguishable.

**Claim 3.9.** *Suppose* $\mathsf{OSARE}_g$ *is secure according to definition 3.1. Then,*

$$\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1.$$

*Proof.* $\mathsf{Hyb}_0$ differs from $\mathsf{Hyb}_1$ only in that in the first $\widehat{z}_0 = \widehat{x}_0 + \widehat{y}_0$ is a real sum encoding, whereas in the second it is a simulated encoding, $\widehat{z}'_0 \leftarrow \mathsf{OSARE}_g.\mathsf{Sim}(1^\lambda, \mathsf{pk}, \mathsf{fsk}_{\mathsf{pct}})$. Accordingly, we can turn any distinguisher between the hybrids to a distinguisher against $\widehat{z}_0$ and $\widehat{z}'_0$, which contradicts the one-sided security of $\mathsf{OSARE}_g$. □

**Claim 3.10.** *Suppose* $\mathsf{OSARE}_e$ *is secure according to definition 3.1. Then,*

$$\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2.$$

*Proof.* $\mathsf{Hyb}_1$ differs from $\mathsf{Hyb}_2$ only in that in the first $\widehat{z}_1 = \widehat{x}_1 + \widehat{y}_1$ is a real sum encoding and in the second it is a simulated encoding, $\widehat{z}'_1 \leftarrow \mathsf{OSARE}_e\mathsf{Sim}(1^\lambda, \mathsf{fpk}, \mathsf{fct})$. Accordingly, we can turn any distinguisher between the hybrids to a distinguisher against $\widehat{z}_1$ and $\widehat{z}'_1$, which contradicts the one-sided security of $\mathsf{OSARE}_e$. □

**Claim 3.11.** *Suppose* $\mathsf{FE}$ *is secure according to definition 2.3. Then,*

$$\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3.$$

*Proof.* $\mathsf{Hyb}_2$ differs from $\mathsf{Hyb}_3$ only in that in the first $(\mathsf{fpk}, \mathsf{fsk}_{\mathsf{pct}}, \mathsf{fct})$ are all sampled according to the real FE algorithms; namely, $(\mathsf{msk}, \mathsf{fpk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$, $\mathsf{fsk}_{\mathsf{pct}} \leftarrow \mathsf{FE.KeyGen}(\mathsf{msk}, \mathsf{pct})$, and $\mathsf{fct} \leftarrow \mathsf{Enc}_{\mathsf{fpk}}((x, \mathsf{sk}))$, whereas in the second they are simulated, $(\mathsf{fpk}', \mathsf{fsk}', \mathsf{fct}') \leftarrow \mathsf{FE.Sim}(1^\lambda, \mathsf{pct}, f(x, y))$. Since the adversary's view in these hybrids is independent of msk and $r_{\mathsf{FE}}$, we can turn any distinguisher between the hybrids to a distinguisher against $(\mathsf{fpk}, \mathsf{fsk}_{\mathsf{pct}}, \mathsf{fct})$ and $(\mathsf{fpk}', \mathsf{fsk}', \mathsf{fct}')$. Since $d((x, \mathsf{sk}), \mathsf{pct}) := f(x, \mathsf{PKE.Dec}_{\mathsf{sk}}(\mathsf{pct})) = f(x, y)$ this contradicts the FE security. □

**Claim 3.12.** *Suppose* PKE *is secure according to definition 2.2. Then,*

$$\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4.$$

*Proof.* $\mathsf{Hyb}_3$ differs from $\mathsf{Hyb}_4$ only in that in the first pct is an encryption of $y$, namely, $\mathsf{pct} \leftarrow \mathsf{PKE.Enc}_{\mathsf{pk}}(y)$, whereas in the second it is an encryption of $0^{n_Y}$, $\mathsf{pct}' \leftarrow \mathsf{PKE.Enc}_{\mathsf{pk}}(0^{n_Y})$. Since the adversary's view in these hybrids is independent of sk and $r_{\mathsf{PKE}}$, we can turn any distinguisher between the hybrids to a distinguisher against pct and pct′ (given pk), which contradicts PKE security. □

Finally, we note that $\mathsf{Hyb}_4$ described the the the simulated output of the constructed ARE. The scheme's security follows. □

# 4 ARE from CDH

In this section, we construct (two-sided) ARE for equality based on the *Computational Diffie Hellman Assumption,* which by [HIKR23, Lemma 5.8] implies ARE for general functions. The resulting construction is rather simple and may be more efficient (and easy to implement) than the construction in the previous section. The presented construction is an adaptation of the pairing-based construction from [HIKR23, Lemma 5.4].

**Groups in Additive Notation.** We denote group encodings in additive notation (c.f. [BHHO08]). We consider a prime-order group $\mathbb{G}$ and identify it with $(\mathbb{Z}_q, +)$. A generator $g \in \mathbb{G}$ will be denoted by $[1]$, every element $g^x$ by $[x]$, and $g^x \times g^y$ by $[x] + [y] = [x + y]$. We also denote multiplication by a scalar $c \in \mathbb{Z}_q$ as $c[x] = [cx]$ to denote exponentiating $g^x$ by $c$ to obtain $g^{cx}$. Let $n$ be the number of bits required to represent any element in $\mathbb{G}$ in binary, and let $\mathsf{bin}(h) \in \{0,1\}^n$ be the binary representation of any element $h \in \mathbb{G}$.

**Inner Product Mod 2.** We denote the inner product mod 2 of two $n$-bit strings $x, y \in \{0,1\}^n$ to be $\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i$ where $x_i$ and $y_i$ are the $i^{th}$ bits of $x$ and $y$.

**Definition 4.1** (Squaring Computational Diffie-Hellman)**.** Let $\mathcal{G}$ be a PPT generator that given $1^\lambda$ outputs $(\mathbb{G}, q, [1])$, where $\mathbb{G}$ is a group of prime order $q$ with generator $[1]$ (and efficient representation and operations). The Squaring-CDH assumption with respect to $\mathcal{G}$ states that for public parameters $(\mathbb{G}, q, [1]) \leftarrow \mathcal{G}(1^\lambda)$, for all polynomial-size circuit families $\{\mathcal{A}_\lambda\}$,

$$\Pr_{x \leftarrow \mathbb{Z}_q}\left[\mathcal{A}(\mathsf{pp}, [x]) = [x^2]\right] = \mathsf{negl}(\lambda).$$

We note that the squaring CDH assumption is known to follow from the (plain) CDH assumption. [MW99, BDZ03].

**Lemma 4.2.** *Assuming Squaring CDH, given* $\mathsf{pp} = (\mathbb{G}, q, [1]) \leftarrow \mathcal{G}(1^\lambda)$*, distinct* $x, y \in [\lfloor q/2 \rfloor]$ *,* $r, s \leftarrow \mathbb{Z}_q, t_1 = r + s$*, and* $t_2 = [rx - sy]$*, then* $t_3 = [r^2x - s^2y]$ *is computationally unpredictable.*

*Proof.* Letting $\tau_1 = r + s$ and $\tau_2 = rx - sy$, we have

$$r = \frac{y\tau_1 + \tau_2}{x + y}$$
$$s = \frac{x\tau_1 - \tau_2}{x + y}.$$

Plugging in values for $r$ and $s$ gives

$$t_3 = [\alpha \cdot \tau_1\tau_2 + \beta \cdot \tau_1^2 + \gamma \cdot \tau_2^2].$$

where

$$\alpha = \frac{4xy}{(x + y)^2}$$
$$\beta = \frac{xy(y - x)}{(x + y)^2}$$
$$\gamma = \frac{x - y}{(x + y)^2}$$

and $\tau_1, \tau_2$ are uniform in $\mathbb{Z}_q$ and independent. Now, we will show that if we have a polynomial-size circuit family $\{\mathcal{A}_\lambda\}$ that given $(\mathsf{pp}, \tau_1, [\tau_2])$ where $(\tau_1, \tau_2) \leftarrow \mathbb{Z}_q^2$ computes $[\alpha \cdot \tau_1\tau_2 + \beta \cdot \tau_1^2 + \gamma \cdot \tau_2^2]$ with non-negligible probability, then we have a polynomial-size circuit family $\{\mathcal{B}_\lambda\}$ which contradicts the Squaring CDH assumption. On input $(\mathsf{pp}, [\rho])$, the adversary $\mathcal{B}$ does the following:

1. Sample $u \leftarrow \mathbb{Z}_q$.
2. Compute $a = \mathcal{A}(\mathsf{pp}, u, [\rho])$.
3. Output $\gamma^{-1}(a - \alpha u[\rho] - \beta u^2[1])$.

If $\mathcal{A}$ outputs the correct result $a = \mathcal{A}(\mathsf{pp}, u, [\rho]) = [\alpha u\rho + \beta u^2 + \gamma\rho^2]$, with non-negligible probability then $\mathcal{B}$ outputs $\gamma^{-1}(a - \alpha u[\rho] - \beta u^2[1]) = [\rho^2]$ with the same probability. □

## 4.1 Construction

We describe a scheme $\mathsf{ARE}_{eq}$ for the equality function over the domain $[d]$ where $(\mathbb{G}, q, [1]) \leftarrow \mathcal{G}(1^\lambda)$ and $q > 2d$.

**Proposition 4.3.** *The scheme* $\mathsf{ARE}_{eq}$ *is correct with probability* $\frac{1}{2}$.

*Proof.* For inputs $x, y \in \mathbb{Z}_q$ and additive encoding $\widehat{z} = (\widehat{z}_0, \widehat{z}_1, \widehat{z}_2, \widehat{z}_3, \widehat{z}_4)$ , we have

$$\begin{aligned}\langle \mathsf{bin}(\widehat{z}_2 - \widehat{z}_0\widehat{z}_1), \widehat{z}_3\rangle &= \langle \mathsf{bin}([R] + [r^2x - s^2y] - [(r + s)(rx - sy)]), S\rangle \\ &= \langle \mathsf{bin}([R] + [rs(x - y)]), S\rangle\end{aligned}$$

- If $x = y$, then the output $\mathsf{Dec}_{\mathsf{pp}}(1^\lambda, \widehat{z}) = \texttt{equal}$ is always correct since we always have $\langle \mathsf{bin}(\widehat{z}_2 - \widehat{z}_0\widehat{z}_1), \widehat{z}_3\rangle = \langle \mathsf{bin}([R]), S\rangle = \widehat{z}_4$.

**Two-Party Equality ARE $\mathsf{ARE}_{eq}$ from CDH**

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

- Set $\mathsf{pp} = (\mathbb{G}, q, [1]) \leftarrow \mathcal{G}(1^\lambda)$.

$\underline{\mathsf{Enc}_{\mathsf{pp}}(1^\lambda, x, 1)}$:

- Sample $r \leftarrow \mathbb{Z}_q$.
- Output $\widehat{x} = (r, [rx], [r^2x], 0, 0)$.

$\underline{\mathsf{Enc}_{\mathsf{pp}}(1^\lambda, y, 2)}$:

- Sample $s, R \leftarrow \mathbb{Z}_q$, and $S \leftarrow \{0,1\}^n$.
- Output $\widehat{y} = (s, [-sy], [-s^2y] + [R], S, \langle \mathsf{bin}([R]), S\rangle)$.

$\underline{\mathsf{Dec}_{\mathsf{pp}}\left(1^\lambda, \widehat{z} = (\widehat{z}_0, \widehat{z}_1, \widehat{z}_2, \widehat{z}_3, \widehat{z}_4)\right)}$:

- If $\langle \mathsf{bin}(\widehat{z}_2 - \widehat{z}_0\widehat{z}_1), \widehat{z}_3\rangle = \widehat{z}_4$, then output `equal`, else output `not-equal`.

Figure 3: Two-Party Equality ARE from CDH

- If $x \neq y$, then we have $\langle \mathsf{bin}([R] + [rs(x-y)]), S\rangle = \langle \mathsf{bin}([R]), S\rangle + \langle a, S\rangle$ for some nonzero $a \in \{0,1\}^n$. The probability of outputting `not-equal` is the probability that $\langle a, S\rangle = 1$, which is $\frac{1}{2}$ because $S$ is sampled independently from $a$.

□

**Proposition 4.4.** *The scheme $\mathsf{ARE}_{eq}$ is computationally secure.*

*Proof.* We define the simulator Sim as follows:

- When inputs are equal, $\mathsf{Sim}(1^\lambda, \texttt{equal})$ randomly samples $r', s', R' \leftarrow \mathbb{Z}_q$, $S' \leftarrow \{0,1\}^n$ and outputs $(\widehat{z}'_0, \widehat{z}'_1, \widehat{z}'_2, \widehat{z}'_3, \widehat{z}'_4) = (r', [s'], [R'], S', \langle \mathsf{bin}([R'] - [r's']), S'\rangle)$.
- When inputs are not equal, $\mathsf{Sim}(1^\lambda, \texttt{not-equal})$ randomly samples $r', s', R' \leftarrow \mathbb{Z}_q, S' \leftarrow \{0,1\}^n, U' \leftarrow \{0,1\}$ and outputs $(\widehat{z}'_0, \widehat{z}'_1, \widehat{z}'_2, \widehat{z}'_3, \widehat{z}'_4) = (r', [s'], [R'], S', U')$.

In both cases, the values $(\widehat{z}_0, \widehat{z}_1, \widehat{z}_2, \widehat{z}_3) = (r+s, [rx - sy], [r^2x - s^2y] + [R], S)$ are independent and uniformly random over their respective domains. Since $x \neq -y$ because $0 < x, y < q/2$, the values $\widehat{z}_0$ and $\widehat{z}_1$ are linearly independent. In addition, since $R$ and $S$ are independent of $r, s, x$, and $y$, we have $\widehat{z}_2$ and $\widehat{z}_3$ are also uniformly random and independent from $\widehat{z}_0$ and $\widehat{z}_1$. Thus, $(\widehat{z}_0, \widehat{z}_1, \widehat{z}_2, \widehat{z}_3) \equiv (\mathcal{U}(\mathbb{Z}_q), \mathcal{U}(\mathbb{G}), \mathcal{U}(\mathbb{G}), \mathcal{U}_n)$.

For the equality case, $\widehat{z}'_4 = \langle \mathsf{bin}([R]), S\rangle$ is equivalent to the distribution of $\widehat{z}'_4$ by the correctness proof, which completes the proof for the equality case.

In the inequality case, from lemma 4.2 we have $\widehat{z}_2 - [R] = [r^2x - s^2y]$ is unpredictable from $(\widehat{z}_0, \widehat{z}_1)$, which implies that $R$ is unpredictable from $(\widehat{z}_0, \widehat{z}_1, \widehat{z}_2)$. Thus, $(\widehat{z}_0, \widehat{z}_1, \widehat{z}_2, \widehat{z}_3, \widehat{z}_4) \approx_c (\mathcal{U}(\mathbb{Z}_q), \mathcal{U}(\mathbb{G}), \mathcal{U}(\mathbb{G}), \mathcal{U}_n, \mathcal{U}_1)$ follows from the Goldreich-Levin hardcore bit theorem [GL89]. This shows the output of Sim is computationally indistinguishable from $\widehat{z}$ in both cases.

□

### 4.2 Correctness Amplification

To amplify the correctness of the scheme, we perform parallel repetition.

**Lemma 4.5.** *For any $\delta \in (0,1)$, $\mathsf{ARE}_{eq}$ can be converted to a scheme $\mathsf{ARE}_{eq,\delta}$ with correctness probability at least $1-\delta$. The communication complexity of $\mathsf{ARE}_{eq,\delta}$ is $O(\log(1/\delta))$ times larger than that of $\mathsf{ARE}_{eq}$.*

*Proof.*

**The scheme $\mathsf{ARE}_{eq,\delta}$.** Let $k = \lceil \log_2(1/\delta) \rceil$. Both parties run $k$ parallel copies of the scheme $\mathsf{ARE}_{eq}$. The evaluator gets $k$ sums $y_1, \ldots, y_k$ from the $k$ copies of $\mathsf{ARE}_{eq}$ and decodes them to get $k$ results $b_i = \mathsf{Dec}_{eq}(y_i)$. It outputs `equal` if $b_i =$ `equal` for all $i \in [k]$ and outputs `not-equal` otherwise.

**Correctness.** If $x = y$, then $\mathsf{ARE}_{eq}$ is perfectly correct, so $\mathsf{ARE}_{eq,\delta}$ will also be perfectly correct. If $x \neq y$, then the probability that $\mathsf{ARE}_{eq,\delta}$ is incorrect is the probability that $b_i =$ `equal` for all $i \in [k]$, which occurs with probability $\left(\frac{1}{2}\right)^k \leq \delta$.

**Security.** Security follows by a standard hybrid argument from the security of $\mathsf{ARE}_{eq}$ since all copies of the scheme are independent of each other. □

## Acknowledgments

N. Bitansky was 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).

## References

[AAP19] Navneet Agarwal, Sanat Anand, and Manoj Prabhakaran. Uncovering algebraic structures in the MPC landscape. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 381–406. Springer, 2019.

[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 $\mathrm{nc}^0$. *SIAM J. Comput.*, 36(4):845–888, 2006.

[App17] Benny Applebaum. Garbled circuits as randomized encodings of functions: a primer. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, pages 1–44. Springer International Publishing, 2017.

[Bar17] Boaz Barak. The complexity of public-key cryptography. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, pages 45–77. Springer International Publishing, 2017.

[BDZ03] Feng Bao, Robert H Deng, and Huafei Zhu. Variations of diffie-hellman problem. In *International Conference on Information and Communications Security*, pages 301–312. Springer, 2003.

[BF24] Nir Bitansky and Sapir Freizeit. Robust additive randomized encodings from IO and pseudo-non-linear codes. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part VIII*, volume 14927 of *Lecture Notes in Computer Science*, pages 109–135. Springer, 2024.

[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.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22(6):644–654, 1976.

[FKN94] Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 554–563. ACM, 1994.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 25–32. ACM, 1989.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 365–377. ACM, 1982.

[HIJ+17] Shai Halevi, Yuval Ishai, Abhishek Jain, Ilan Komargodski, Amit Sahai, and Eylon Yogev. Non-interactive multiparty computation without correlated randomness. *IACR Cryptol. ePrint Arch.*, page 871, 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.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 294–304. IEEE Computer Society, 2000.

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan J. Eidenbenz, and Ricardo Conejo, editors, *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings*, volume 2380 of *Lecture Notes in Computer Science*, pages 244–256. Springer, 2002.

[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.

[Ish13] Yuval Ishai. Randomization techniques for secure computation. In Manoj Prabhakaran and Amit Sahai, editors, *Secure Multi-Party Computation*, volume 10 of *Cryptology and Information Security Series*, pages 222–248. IOS Press, 2013.

[MW99] Ueli M. Maurer and Stefan Wolf. The relationship between breaking the diffie-hellman protocol and computing discrete logarithms. *SIAM J. Comput.*, 28(5):1689–1721, 1999.

[RSA83] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems (reprint). *Commun. ACM*, 26(1):96–99, 1983.

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 463–472. ACM, 2010.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167. IEEE Computer Society, 1986.