# Functional Encryption for Attribute-Weighted Sums from $k$-Lin

Michel Abdalla[1], Junqing Gong[2(✉)], and Hoeteck Wee[1,3]

[1] CNRS, ENS and PSL, Paris, France
michel.abdalla@ens.fr, wee@di.ens.fr
[2] East China Normal University, Shanghai, China
jqgong@sei.ecnu.edu.cn
[3] NTT Research, Palo Alto, CA, USA

**Abstract.** We present functional encryption schemes for attribute-weighted sums, where encryption takes as input $N$ attribute-value pairs $(x_i, z_i)$ where $x_i$ is public and $z_i$ is private; secret keys are associated with arithmetic branching programs $f$, and decryption returns the weighted sum $\sum_{i=1}^{N} f(x_i)z_i$ while leaking no additional information about the $z_i$'s. Our main construction achieves

**(1)** compact public parameters and key sizes that are independent of $N$ and the secret key can decrypt a ciphertext for any a-priori unbounded $N$;
**(2)** short ciphertexts that grow with $N$ and the size of $z_i$ but not $x_i$;
**(3)** simulation-based security against unbounded collusions;
**(4)** relies on the standard $k$-linear assumption in prime-order bilinear groups.

## 1 Introduction

In this work, we consider the problem of computing aggregate statistics on encrypted databases. Consider a database of $N$ attribute-value pairs $(x_i, z_i)_{i=1,\ldots,N}$, where $x_i$ is a public attribute of user $i$ (e.g. demographic data), and $z_i$ is private sensitive data associated with user $i$ (e.g. salary, medical condition, loans, college admissions outcome). Given a function $f$, we want to privately compute weighted sums over the $z_i$'s corresponding to

$$\sum_{i=1}^{N} f(x_i)z_i$$

We refer to this quantity as an *attribute-weighted sum*. An important special case is when $f$ is a boolean predicate, so that the attribute-weighted sum

$$\sum_{i=1}^{N} f(x_i)z_i = \sum_{i:f(x_i)=1} z_i \tag{1}$$

M. Abdalla—Supported by ERC Project aSCEND (H2020 639554) and the French FUI project ANBLIC.
J. Gong—Supported by NSFC-ISF Joint Scientific Research Program (61961146004) and the ERC Project aSCEND (H2020 639554). Part of this work was done while at ENS, Paris.
H. Wee—Supported in part by ERC Project aSCEND (H2020 639554).


D. Micciancio and T. Ristenpart (Eds.): CRYPTO 2020, LNCS 12170, pp. 685–716, 2020.
https://doi.org/10.1007/978-3-030-56784-2_23

corresponds to the average $z_i$ over all users whose attribute $x_i$ satisfies the predicate $f$. Concrete examples include average salaries of minority groups holding a particular job title ($z_i$ = salary) and approval ratings of an election candidate amongst specific demographic groups in a particular state ($z_i$ = rating). Similarly, if $z_i$ is boolean, then the attribute-weighted sum becomes $\sum_{i:z_i=1} f(x_i)$. This could capture for instance the number of and average age of smokers with lung cancer ($z_i$ = lung cancer).

**This work.** We study functional encryption (FE) schemes for attribute-weighted sums [13,24,26,36], for a more general setting where the attribute-value pairs and the output of $f$ are vectors. That is, we would like to encrypt $N$ attribute-value pairs $(\mathbf{x}_i, \mathbf{z}_i)_{i=1,\ldots,N}$ to produce a ciphertext $\mathsf{ct}$, and generate secret keys $\mathsf{sk}_f$ so that decrypting $\mathsf{ct}$ with $\mathsf{sk}_f$ returns the attribute-weighted sum $\sum_i f(\mathbf{x}_i)^\top \mathbf{z}_i$ while leaking no additional information about the individual $\mathbf{z}_i$'s. We want to support rich and expressive functions $f$, such as boolean formula and simple arithmetic computation. In addition, we want simulation-based security against collusions, so that an adversary holding secret keys for different functions learns nothing about the $\mathbf{z}_i$'s beyond the attribute-weighted sums for all of these functions.

In many databases, it is often the case that the size of each attribute-value pair $(\mathbf{x}_i, \mathbf{z}_i)$ is small and a-priori bounded, whereas the number of *slots* $N$ is large and a-priori *unbounded*. This motivates the notion of an *unbounded-slot* FE scheme for attribute-weighted sums, where a secret key $\mathsf{sk}_f$ can decrypt encrypted databases with an arbitrary number of slots. Indeed, handling arbitrary-sized inputs is also the motivation behind studying ABE and FE schemes for DFA and NFA [7,38]. In an unbounded-slot FE, key generation and the size of $\mathsf{sk}_f$ depends only on $f$ and not $N$. This provides stronger flexibility than standard ABE and FE (even in the so-called unbounded setting [14,19,25,32]), where each $\mathsf{sk}_f$ only works for a fixed $N$. In practice, this means that we can reuse the same set-up and secret keys across multiple databases without an a-priori upper bound on the database size $N$.

### 1.1 Our Results

We present an *unbounded-slot* functional encryption scheme for attribute-weighted sums for the class of functions $f$ captured by arithmetic branching programs (ABP), a powerful model of computation that captures both boolean formula and branching programs with only a linear blow-up in size. Our construction achieves:

(1) compact public parameters and key sizes that are independent of $N$;
(2) short ciphertexts that grow with $N$ and the size of $\mathbf{z}_i$ but not $\mathbf{x}_i$;
(3) selective[1], simulation-based security against unbounded collusions;
(4) relies on the standard $k$-linear assumption in prime-order bilinear groups.

[1] We actually achieve semi-adaptive security [16], a slight strengthening of selective security.

As with all prior FE schemes that rely on DDH and bilinear groups [1,3,6,10,17,28,29,33], efficient decryption requires that the output of the computation $\sum_{i=1}^{N} f(\mathbf{x}_i)^\top \mathbf{z}_i$ lies in a polynomial-size domain. We also show how to extend our unbounded-slot scheme to a setting where the database is distributed across multiple clients that do not completely trust one another [18,21], assuming some simple non-interactive MPC set-up amongst the clients that does not depend on the database and does not require interaction with the key authority.

**Prior works.** While we regard the unbounded-slot setting as the key conceptual and technical novelty of this work, we note that FE for attribute-weighted sums for $N = 1$ already captures many functionalities considered in the literature, e.g.

(i) FE for inner product [1,6] where $f$ outputs a fixed vector,
(ii) attribute-based encryption (ABE) by taking $z$ to be the payload,
(iii) attribute-based inner-product FE [2,17], where ciphertexts are associated with a public $\mathbf{x}$ and a private $\mathbf{z}$, and keys with a boolean formula $g$ and a vector $\mathbf{y}$, and decryption returns $\mathbf{z}^\top \mathbf{y}$ iff $g(\mathbf{x}) = 1$, by taking $f(\mathbf{x}) := \mathbf{y} \cdot g(\mathbf{x})$, which can be computed using an ABP.

On the other hand, none of these three classes captures the special case of attribute-weighted sums in (1). We show a comparison in Fig. 1. The more recent works in [28,29] do capture a larger class supporting quadratic instead of linear functions over $\mathbf{z}$,[2] but in a weaker secret-key setting with indistinguishability-based security, which is nonetheless sufficient for the application to obfuscation. As articulated [13], simulation-based security is the right notion for functional encryption applied to real-world data. Finally, none of these works consider the unbounded-slot setting.

### 1.2 Our Construction

We present a high-level overview of our unbounded-slot FE scheme for attribute-weighted sums. We start with a one-slot scheme that only handles $N = 1$, and then "bootstrap" to the unbounded-slot setting. The main technical novelty of this work lies in the bootstrapping, which is what we would focus on in this section.

**A one-slot scheme.** In a one-slot FE scheme, we want to encrypt $(\mathbf{x}, \mathbf{z})$ and generate secret keys $\mathsf{sk}_f$ for computing $f(\mathbf{x})^\top \mathbf{z}$, while leaking no additional information about $\mathbf{z}$. We adopt the framework of Wee's [40] (which in turn builds on [27,30,37,39]) that builds a FE scheme for a closely related functionality $f(\mathbf{x})^\top \mathbf{z} \stackrel{?}{=} 0$; the construction also achieves selective, simulation-based security under the $k$-Lin assumption in prime-order bilinear groups. We achieve a smaller

[2] Note that we can also capture the same class with a quadratic blow-up in ciphertext size.

ciphertext, and an algebraically more concise and precise description. Our simulator also embeds the output of the ideal functionality $f(\mathbf{x})^\top \mathbf{z}$ into the simulated $\mathsf{sk}_f$. This is in some sense inherent for two reasons: (i) the ciphertext has a fixed size and cannot accommodate an a-priori unbounded number of key queries [4], (ii) in the selective setting, we do not know $f$ or $f(\mathbf{x})^\top \mathbf{z}$ while simulating the ciphertext.

**The unbounded-slot scheme.** A very natural approach is to use the one-slot scheme to compute

$$f(\mathbf{x}_i)^\top \mathbf{z}_i, i = 1, 2, \ldots, N \tag{2}$$

by providing $N$ independent encryptions $\mathsf{ct}_{\mathbf{x}_i,\mathbf{z}_i}$ of $(\mathbf{x}_i, \mathbf{z}_i)$. The secret key is exactly that for the one-slot scheme and therefore independent of $N$, and decryption proceeds by decrypting each of the $N$ one-slot ciphertexts, and then computing their sum. The only problem with this approach is that it is insecure since decryption leaks the intermediate summands.

*First idea.* To avoid this leakage, we would computationally mask the summands using DDH tuples, by using the one-slot scheme to compute

$$[f(\mathbf{x}_i)^\top \mathbf{z}_i + w_i r], \; i = 1, 2, \ldots, N \tag{3}$$

where

- the $w_i$'s are sampled during encryption subject to the constraint $\sum_{i=1}^{N} w_i = 0$;
- $r$ is fresh per secret key; and
- $[\cdot]$ denotes "in the exponent" of a bilinear group.

Multiplying the partial decryptions yields $[\sum_i f(\mathbf{x}_i)^\top \mathbf{z}_i]$, and we need to perform a brute-force discrete log to recover the answer. Indeed, we can modify the one-slot scheme to support the functionality in (3), where the one-slot encryption takes as input $(\mathbf{x}_i, \mathbf{z}_i \| w_i)$ (where $w_i$ is also private) to produce a ciphertext $\mathsf{ct}_{\mathbf{x}_i,\mathbf{z}_i\|w_i}$, and with secret keys $\mathsf{sk}_{f,r}$ associated with $(f, r)$. Henceforth, we describe the proof strategy for a single secret key query for simplicity, but everything we describe extends quite readily to an unbounded number of key queries.

The intuition is that the partial decryptions now yield

$$\begin{array}{rlllll}
 & (\ \mathsf{Dec}(\mathsf{sk}_{f,r}, \mathsf{ct}_{\mathbf{x}_1,\mathbf{z}_1\|w_1}), & \mathsf{Dec}(\mathsf{sk}_{f,r}, \mathsf{ct}_{\mathbf{x}_2,\mathbf{z}_2\|w_2}), & \ldots, \mathsf{Dec}(\mathsf{sk}_{f,r}, \mathsf{ct}_{\mathbf{x}_N,\mathbf{z}_N\|w_N}) & ) \\
= & (\ [f(\mathbf{x}_1)^\top \mathbf{z}_1 + w_1 r], & [f(\mathbf{x}_2)^\top \mathbf{z}_2 + w_2 r], & \ldots, [f(\mathbf{x}_N)^\top \mathbf{z}_N + w_N r] & ), \\
\overset{\text{DDH}}{\approx_c} & (\ [f(\mathbf{x}_1)^\top \mathbf{z}_1 + w'_1], & [f(\mathbf{x}_2)^\top \mathbf{z}_2 + w'_2], & \ldots, [f(\mathbf{x}_N)^\top \mathbf{z}_N + w'_N] & ), \ \textstyle\sum w'_i = 0 \\
\approx_s & (\ [\textstyle\sum_i f(\mathbf{x}_i)^\top \mathbf{z}_i + w'_1], & [w'_2], & \ldots, [w'_N] & ),
\end{array}$$

As with the one-slot scheme, we need to embed these $N$ partial descriptions into $\mathsf{sk}_{f,r}$ in the proof of security. Translating this intuition into a proof would then require embedding $\approx N$ units of statistical entropy into the simulated $\mathsf{sk}_{f,r}$ in the final game; this means that the size of $\mathsf{sk}_{f,r}$ would grow with $N$, which we want to avoid!

*Second idea.* Instead, we will do a hybrid argument over the $N$ slots, collecting "partial sums" $\sum_{i \le \eta} f(\mathbf{x}_i)^\top \mathbf{z}_i$ (with $1 \le \eta \le N$) as we go along, which we then embed into the simulated $\mathsf{sk}_{f,r}$. This proof strategy is in fact inspired by proof techniques introduced in the recent ABE for DFA from $k$-Lin [22], notably the idea of propagating entropy along the execution path of a DFA.

In particular, for $N = 3$, partial decryption now yields

$$\begin{array}{rl} & (\ \mathsf{Dec}(\mathsf{sk}_{f,r}, \mathsf{ct}_{\mathbf{x}_1,\mathbf{z}_1\|w_1}), \quad \mathsf{Dec}(\mathsf{sk}_{f,r}, \mathsf{ct}_{\mathbf{x}_2,\mathbf{z}_2\|w_2}), \quad \mathsf{Dec}(\mathsf{sk}_{f,r}, \mathsf{ct}_{\mathbf{x}_3,\mathbf{z}_3\|w_3})\ ) \\ = & (\ [f(\mathbf{x}_1)^\top \mathbf{z}_1 + w_1 r], \quad [f(\mathbf{x}_2)^\top \mathbf{z}_2 + w_2 r], \quad [f(\mathbf{x}_3)^\top \mathbf{z}_3 + w_3 r]\ ) \\ \overset{\text{DDH}}{\approx_c} & (\ [f(\mathbf{x}_1)^\top \mathbf{z}_1 + f(\mathbf{x}_2)^\top \mathbf{z}_2 + w_1 r], \quad [w_2 r], \quad [f(\mathbf{x}_3)^\top \mathbf{z}_3 + w_3 r]\ ) \\ \overset{\text{DDH}}{\approx_c} & (\ [f(\mathbf{x}_1)^\top \mathbf{z}_1 + f(\mathbf{x}_2)^\top \mathbf{z}_2 + f(\mathbf{x}_3)^\top \mathbf{z}_3 + w_1 r], \quad [w_2 r], \quad [w_3 r]\ ) \end{array} \tag{4}$$

where the first $\overset{\text{DDH}}{\approx_c}$ uses pseudorandomness of $([w_2 r], [r])$ and the second uses that of $([w_3 r], [r])$.

Next, we need to design the ciphertext and key distributions for the unbounded-slot scheme so that partial decryption yields the quantities in (4). We begin by defining the final simulated ciphertext-key pair as follows:

$$(\mathsf{ct}^*_{\mathbf{x}_1}, \mathsf{ct}_{\mathbf{x}_2,\mathbf{0}\|w_2}, \ldots, \mathsf{ct}_{\mathbf{x}_N,\mathbf{0}\|w_N}), \quad \mathsf{sk}^*_{f,r} \tag{5}$$

where

- $(\mathsf{ct}^*_{\mathbf{x}_1}, \mathsf{sk}^*_{f,r})$ are obtained using the simulator for the one-slot scheme so that

$$\mathsf{Dec}(\mathsf{sk}^*_{f,r}, \mathsf{ct}^*_{\mathbf{x}_1}) = [w_1 r + \sum\nolimits_i f(\mathbf{x}_i)^\top \mathbf{z}_i]$$

  That is, we embed $[w_1 r + \sum_i f(\mathbf{x}_i)^\top \mathbf{z}_i]$ into the simulated $\mathsf{sk}^*_{f,r}$;
- $\mathsf{ct}_{\mathbf{x}_i,\mathbf{0}\|w_i}, i > 1$ are generated as normal encryptions of $(\mathbf{x}_i, \mathbf{0}\|w_i)$ (instead of normal encryptions of $(\mathbf{x}_i, \mathbf{z}_i\|w_i)$) so that

$$\mathsf{Dec}(\mathsf{sk}^*_{f,r}, \mathsf{ct}_{\mathbf{x}_i,\mathbf{0}\|w_i}) = \mathsf{Dec}(\mathsf{sk}_{f,r}, \mathsf{ct}_{\mathbf{x}_i,\mathbf{0}\|w_i}) = [w_i r], i > 1$$

  Here, we use fact that simulated secret keys behave like normal secret keys when used to decrypt normal ciphertexts.

This distribution can be computed given just $\sum_i f(\mathbf{x}_i)^\top \mathbf{z}_i$ and matches exactly what we need in the final game in (4).

*Third idea.* Now, consider the following attempt to interpolate between the normal distributions and the simulated distributions for the case $N = 2$:

$$\begin{array}{rlll} & (\ \mathsf{ct}_{\mathbf{x}_1,\mathbf{z}_1\|w_1}, & \mathsf{ct}_{\mathbf{x}_2,\mathbf{z}_2\|w_2}, & \mathsf{sk}_{f,r}\ ) \\ \approx_c & (\ \mathsf{ct}^*_{\mathbf{x}_1}, & \mathsf{ct}_{\mathbf{x}_2,\mathbf{z}_2\|w_2}, & \mathsf{sk}^*_{f,r}\ ), \quad \mathsf{Dec}(\mathsf{sk}^*_{f,r}, \mathsf{ct}^*_{\mathbf{x}_1}) = [f(\mathbf{x}_1)^\top \mathbf{z}_1 + w_1 r] \\ \approx_c & (\ \mathsf{ct}^*_{\mathbf{x}_1}, & ???, & \mathsf{sk}^*_{f,r}\ ), \\ \approx_c & (\ \mathsf{ct}^*_{\mathbf{x}_1}, & \mathsf{ct}_{\mathbf{x}_2,\mathbf{0}\|w_2}, & \mathsf{sk}^*_{f,r}\ ), \quad \mathsf{Dec}(\mathsf{sk}^*_{f,r}, \mathsf{ct}^*_{\mathbf{x}_1}) = [f(\mathbf{x}_1)^\top \mathbf{z}_1 + f(\mathbf{x}_2)^\top \mathbf{z}_2 + w_1 r] \end{array}$$

where the first row is the real distribution, the last row is the simulated distribution in (5), and the first $\approx_c$ follows from simulation-based security of the one-slot scheme. A natural idea is to replace "???" with a simulated ciphertext $\mathsf{ct}^*_{\mathbf{x}_2}$ but this is problematic for two reasons: first, we cannot switch between a

normal and simulated ciphertext in the presence of a simulated key, and second, the simulator can only generate a single simulated ciphertext.

Luckily, we can overcome both difficulties by modifying the unbounded-slot FE scheme to use *two* independent copies of the one-slot scheme as follows:

- setup generates two one-slot master public-secret key pairs $(\mathsf{mpk}_1, \mathsf{msk}_1)$, $(\mathsf{mpk}_2, \mathsf{msk}_2)$;
- to encrypt $(\mathbf{x}_i, \mathbf{z}_i)_{i=1,\ldots,N}$, we generate $\mathsf{ct}_{\mathbf{x}_1, \mathbf{z}_1 \| w_1}$ w.r.t $\mathsf{mpk}_1$ and the remaining $\mathsf{ct}_{\mathbf{x}_i, \mathbf{z}_i \| w_i}, i = 2, \ldots, N$ w.r.t. $\mathsf{mpk}_2$;
- the secret key contains two one-slot secret keys $\mathsf{sk}_{f,r,1}, \mathsf{sk}_{f,r,2}$ generated for $(f, r)$ but using $\mathsf{msk}_1, \mathsf{msk}_2$ respectively.

That would in fact be our final construction, where the asymmetry of encryption with respect to the first slot reflects the asymmetry of the simulated ciphertext in (5). Note that the first issue goes away because we can switch between a normal and simulated ciphertext w.r.t. $\mathsf{mpk}_2$ in the presence of a simulated secret key w.r.t. $\mathsf{mpk}_1$; the second goes away because the two simulated ciphertext correspond to $\mathsf{mpk}_1$ and $\mathsf{mpk}_2$ respectively. We defer the remaining details to the technical overview in Sect. 2 and the formal scheme in Sect. 7.

| **Scheme** | Enc | KeyGen | **Function** | **Security** | $\lvert\mathsf{ct}\rvert$ |
|---|---|---|---|---|---|
| OT12, KSW08 [30,34,35] | $\mathbf{z}$ | $\mathbf{y}$ | $\mathbf{z}^\top \mathbf{y} \stackrel{?}{=} 0$ | AD-IND | $O(\lvert\mathbf{z}\rvert)$ |
| ALS16, ABDP15 [1,6] | $\mathbf{z}$ | $\mathbf{y}$ | $\mathbf{z}^\top \mathbf{y}$ | AD-IND | $O(\lvert\mathbf{z}\rvert)$ |
| W17 [40] | $\mathbf{x}, \mathbf{z}$ | $f$ ABP | $\mathbf{z}^\top f(\mathbf{x}) \stackrel{?}{=} 0$ | SA-SIM | $O(\lvert\mathbf{x}\rvert + \lvert\mathbf{z}\rvert)$ |
| DOT18 [19] | $\mathbf{x}, \mathbf{z}$ | $f$ ABP | $\mathbf{z}^\top f(\mathbf{x}) \stackrel{?}{=} 0$ | AD-SIM | $O(\lvert\mathbf{x}\rvert + \lvert\mathbf{z}\rvert)$ |
| ACGU20, CZY19 [2,17] | $\mathbf{x}, \mathbf{z}$ | $\mathbf{y}, f$ NC1 | $f(\mathbf{x}) \cdot \mathbf{z}^\top \mathbf{y}$ | AD-IND | $O(\lvert\mathbf{x}\rvert + \lvert\mathbf{z}\rvert)$ |
| ACGU20 [2] | $\mathbf{z}_1, \mathbf{z}_2$ | $\mathbf{y}_1, \mathbf{y}_2$ | $\mathbf{z}_1^\top \mathbf{y}_1$ if $\mathbf{z}_2^\top \mathbf{y}_2 = 0$ | AD-IND | $O(\lvert\mathbf{z}_1\rvert + \lvert\mathbf{z}_2\rvert)$ |
| This work (§5) | $\mathbf{x}, \mathbf{z}$ | $f$ ABP | $\mathbf{z}^\top f(\mathbf{x})$ | SA-SIM | $O(\lvert\mathbf{z}\rvert)$ |

**Fig. 1.** Comparison of prior public-key schemes with our construction for $N = 1$. Throughout, $\mathbf{x}$ is public and $\mathbf{z}, \mathbf{z}_1, \mathbf{z}_2$ are private, and $\lvert\mathsf{ct}\rvert$ omits the contribution from $\mathbf{x}$.

**The multi-client setting.** Now, consider a setting where the database $(\mathbf{x}_i, \mathbf{z}_i)_{i=1,\ldots,N}$ are distributed across multiple clients that do not completely trust one another [18,21]; in practice, the clients could correspond to hospitals holding medical records for different patients, or colleges holding admissions data. It suffices to just consider the setting with $N$ clients where client $i$ holds $(\mathbf{x}_i, \mathbf{z}_i)$. Note that to produce the ciphertext in our unbounded-slot FE scheme, it suffices for the $N$ clients to each hold a random private $w_i$ (per database) subject to the constraint $\sum w_i = 0$, which is simple to generate via a non-interactive MPC protocol where each client sends out additive shares of 0 [11]. Moreover, generating the $w_i$'s can take place in an offline, pre-processing phase before knowing the database, and does not require interacting with the key generation authority. Moreover, our unbounded-slot FE scheme also achieves a meaningful

notion of security, namely that if some subset $S$ of clients collude and additionally learn some $\mathsf{sk}_f$, they will not learn anything about the remaining $\mathbf{z}_i$'s apart from $\sum_{i \notin S} f(\mathbf{x}_i)^\top \mathbf{z}_i$ (that is, the attribute-weighted sum as applied to the honest clients' inputs); security is simulation-based and also extends to the many-key setting. In order to achieve this, we require a slight modification to the scheme to break the asymmetry with respect to the first slot: to encrypt $(\mathbf{x}_i, \mathbf{z}_i)$, client $i$ samples random $\mathbf{z}'_i, w'_i$ and publishes a one-slot encryption of $(\mathbf{x}_i, \mathbf{z}'_i \| w'_i)$ under $\mathsf{mpk}_1$ and another of $(\mathbf{x}_i, \mathbf{z} - \mathbf{z}'_i \| w_i - w'_i)$ under $\mathsf{mpk}_2$. This readily gives us a multi-client unbounded-slot FE for attribute-weighted sums; we refer the reader to full paper for more details of the definition, construction and proof.

### 1.3 Discussion

**Additional related works.** As noted earlier in the introduction, our unbounded-slot notion is closely related to uniform models of computation with unbounded input lengths, such as ABE and FE for DFA and NFA [7,8,22,38]. At a very high level, our construction may be viewed as following the paradigm in [7,8] for building ABE/FE for uniform models of computation by "stitching" together ABE/FE for the smaller step functions; in our setting, the linear relation between the step functions and the overall computation makes "stitching" much simpler. The way we use two copies of the one-slot scheme is also analogous to the "two-slot, interweaving dual system encryption" argument used in the previous ABE for DFA from $k$-Lin in [22], except our implementation is simpler and more modular.

**On selective vs adaptive security.** We believe that selective, simulation-based security already constitutes a meaningful notion of security for many of the applications we have in mind. For instance, in medical studies, medical records and patient conditions (the $\mathbf{x}_i, \mathbf{z}_i$'s) will not depend –not in the short run, at least– adaptively on the correlations (the functions $f$'s) that researchers would like to investigate. Nonetheless, we do agree that extending our results to achieve adaptive security is an important research direction. Concretely,

- Can we show that the one-slot scheme achieves simulation-based, adaptive security in the generic group model, as has been shown for a large class of selectively secure ABEs [9]?
- Can we construct an adaptively secure unbounded-slot FE for arithmetic branching programs with compact ciphertexts without the one-use restriction from $k$-Lin? We conjecture that our transformation from one-slot to unbounded-slot preserves adaptive security. Solving the one-slot problem would require first adapting the techniques for adaptive simulation-based security in [5,19], and more recent advances in [31] to avoid the one-use restriction.

**Open problems.** We conclude with two other open problems. One is whether we can construct (one-slot) FE for attribute-weighted sums from LWE, simultaneously generalizing prior ABE and IPFE schemes from LWE [6,12,23]; an affirmative solution would likely also avoid the polynomial-size domain limitation. Another is to achieve stronger notions of security for the multi-client setting where the $w_i$'s could be reused across multiple databases.

**Organization.** We provide a more detailed technical overview in Sect. 2. We present preliminaries, definitions and tools in Sects. 3 and 4. We present our one-slot scheme and an extension in Sects. 5 and 6, and the unbounded-slot scheme in Sect. 7.

## 2 Technical Overview

We proceed with a more technical overview of our construction, building on the overview given in Sect. 1.2, and giving more details on the one-slot scheme. We summarize the parameters of the one-slot and unbounded-slot scheme in Fig. 2.

### 2.1 One-Slot Scheme

*Notation.* We will make extensive use of tensor products. For instance, we will write the linear function $x_1\mathbf{U}_1 + x_2\mathbf{U}_2$ as

$$(\mathbf{U}_1 \| \mathbf{U}_2) \begin{pmatrix} x_1\mathbf{I} \\ x_2\mathbf{I} \end{pmatrix} = (\mathbf{U}_1 \| \mathbf{U}_2) \left( \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \otimes \mathbf{I} \right)$$

| Scheme | $\lvert\mathsf{ct}\rvert$ | $\lvert\mathsf{sk}\rvert$ | Assumption |
|---|---|---|---|
| $\Pi_{\mathsf{one}}$ (§ 5) | $n' + 2k + 1$ | $(k+1)nm + (2k+1)m + (k+1)n'$ | $k$-Lin |
| | $n' + 3$ | $2nm + 3m + 2n'$ | SXDH |
| $\Pi_{\mathsf{ubd}}$ (§ 7) | $n'N + (3k+1)N$ | $(2k+2)nm + (4k+2)m + (2k+2)n' + k$ | $k$-Lin |
| | $n'N + 4N$ | $4nm + 6m + 4n' + 1$ | SXDH |

**Fig. 2.** Summary of ciphertext and key sizes of our one-slot scheme $\Pi_{\mathsf{one}}$ and unbounded-slot scheme $\Pi_{\mathsf{ubd}}$. Recall that $n = |\mathbf{x}| = |\mathbf{x}_i|$, $n' = |\mathbf{z}| = |\mathbf{z}_i|$, $m$ is proportional to the size of $f$ and $N$ is the number of slots. In the table, we count the number of group elements in $\mathbb{G}_1$ (resp. $\mathbb{G}_2$) in the column $|\mathsf{ct}|$ (resp. column $|\mathsf{sk}|$). Note that SXDH = 1−Lin.

This allows us to concisely and precisely capture "compilers" where we substitute scalars with matrices, as well as the underlying linear relations, which may refer to left or right multiplication, and act on scalars or matrices.

**Partial garbling.** Recall the starting point for ABE for ABP as an "arithmetic secret-sharing scheme" that on input an ABP $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$ and a secret $z \in \mathbb{Z}_p$, outputs $m$ affine functions $\ell_1, \ldots, \ell_m : \mathbb{Z}_p^n \to \mathbb{Z}_p$ such that for all $\mathbf{x} \in \mathbb{Z}_p^n$:

- (correctness) given $\ell_1(\mathbf{x}), \ldots, \ell_m(\mathbf{x})$ along with $f, \mathbf{x}$, we can recover $z$ if $f(\mathbf{x}) \neq 0$.
- (privacy) given $\ell_1(\mathbf{x}), \ldots, \ell_m(\mathbf{x})$ along with $f, \mathbf{x}$, we learn nothing about $z$ if $f(\mathbf{x}) = 0$.

In particular, the coefficients of the functions $\ell_1, \ldots, \ell_m$ depends linearly on the randomness used in secret sharing.

Partial garbling generalizes the above as follows: on input an ABP $f : \mathbb{Z}_p^n \to \mathbb{Z}_p^{n'}$, outputs $m+1$ affine functions $\ell_0, \ell_1, \ldots, \ell_m$ such that for all $\mathbf{x} \in \mathbb{Z}_p^n, \mathbf{z} \in \mathbb{Z}_p^{n'}$:

- (correctness) given $\ell_0(\mathbf{z}), \ell_1(\mathbf{x}), \ldots, \ell_m(\mathbf{x})$ along with $f, \mathbf{x}$, we can recover $f(\mathbf{x})^\top \mathbf{z}$.
- (privacy) given $\ell_0(\mathbf{z}), \ell_1(\mathbf{x}), \ldots, \ell_m(\mathbf{x})$ along with $f, \mathbf{x}$, we learn nothing about $\mathbf{z}$ apart from $f(\mathbf{x})^\top \mathbf{z}$.

Henceforth, we will use $\mathbf{t}^\top(\mathbf{L}_1(\mathbf{x} \otimes \mathbf{I}_m) + \mathbf{L}_0) \in \mathbb{Z}_p^m$ to denote the $m$ linear functions $\ell_1(\mathbf{x}), \ldots, \ell_m(\mathbf{x})$,[3] where $\mathbf{t} \leftarrow \mathbb{Z}_p^{m+n'-1}$ corresponds to the randomness used in the secret sharing; $\mathbf{L}_1 \in \mathbb{Z}_p^{(m+n'-1)\times mn}, \mathbf{L}_0 \in \mathbb{Z}_p^{(m+n'-1)\times m}$ depends only on the function $f$, and $m$ is linear in the size of the ABP $f$.

**Basic scheme.** We rely on an asymmetric bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ of prime order $p$ where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. We use $[\cdot]_1, [\cdot]_2, [\cdot]_T$ to denote component-wise exponentiations in respective groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ [20]. Our starting point is the following scheme[4]:

$$\begin{aligned} \mathsf{mpk} &= \big( [\mathbf{w}]_1, [\mathbf{u}]_1, [v]_1 \big) \quad \text{and} \quad \mathsf{msk} = \big( \mathbf{w}, \mathbf{u}, v \big) \\ \mathsf{ct}_{\mathbf{x},\mathbf{z}} &= \big( [s]_1, [\mathbf{z} + s\mathbf{w}]_1, [s(\mathbf{u}^\top \mathbf{x} + v)]_1 \big) \in \mathbb{G}_1^{n'+2} \\ \mathsf{sk}_f &= \big( [\underline{\mathbf{t}} + \mathbf{w}]_2, [\mathbf{t}^\top \mathbf{L}_1 + \mathbf{u}^\top (\mathbf{I}_n \otimes \mathbf{r}^\top)]_2, [\mathbf{t}^\top \mathbf{L}_0 + v\mathbf{r}^\top]_2, [\mathbf{r}]_2 \big) \end{aligned} \tag{6}$$

where

$$\mathbf{w} \leftarrow \mathbb{Z}_p^{n'}, \mathbf{u} \leftarrow \mathbb{Z}_p^n, v \leftarrow \mathbb{Z}_p, \mathbf{t} \leftarrow \mathbb{Z}_p^{m+n'-1}, \mathbf{r} \leftarrow \mathbb{Z}_p^m$$

[3] As an example with $n = 2, m = 3$, we have

$$\begin{aligned} &\big(a_{11}x_1 + a_{12}x_2 + b_1,\, a_{21}x_1 + a_{22}x_2 + b_2,\, a_{31}x_1 + a_{32}x_2 + b_3\big) \\ &= (a_{11}, a_{21}, a_{31}, a_{12}, a_{22}, a_{32})\Big(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \otimes \mathbf{I}_3\Big) + (b_1, b_2, b_3). \end{aligned}$$

[4] The scheme in [40] has a larger ciphertext of the form: $\mathsf{ct}_{\mathbf{x},\mathbf{z}} = \big( [s]_1, [\mathbf{z} + s\mathbf{w}]_1, [s(\mathbf{u} + v\mathbf{x})]_1 \big) \in \mathbb{G}_1^{n+n'+1}$.

Decryption uses the fact that

$$\mathbf{t}^\top(\mathbf{L}_1(\mathbf{x}\otimes\mathbf{I}_m)+\mathbf{L}_0) = \\ (\mathbf{t}^\top\mathbf{L}_1+\mathbf{u}^\top(\mathbf{I}_n\otimes\mathbf{r}^\top))\cdot(\mathbf{x}\otimes\mathbf{I}_m)+(\mathbf{t}^\top\mathbf{L}_0+v\mathbf{r}^\top)-(\mathbf{u}^\top\mathbf{x}+v)\cdot\mathbf{r}^\top \quad (7)$$

which in turn uses $(\mathbf{I}_n\otimes\mathbf{r}^\top)\cdot(\mathbf{x}\otimes\mathbf{I}_m)=\mathbf{x}\cdot\mathbf{r}^\top$. Using the pairing and the above relation, we can recover

$$[\mathbf{z}-s\underline{\mathbf{t}}]_T, [s\mathbf{t}^\top(\mathbf{L}_1(\mathbf{x}\otimes\mathbf{I}_m)+\mathbf{L}_0)]_T$$

We can then apply reconstruction "in the exponent" to recover $[f(\mathbf{x})^\top\mathbf{z}]_T$ and thus $f(\mathbf{x})^\top\mathbf{z}$ via brute-force DLOG.

**Security in the secret-key setting.** The scheme as written already achieves simulation-based selective security in the secret-key, many-key setting (that is, against an adversary that does not see $\mathsf{mpk}$); this holds under the DDH assumption in $\mathbb{G}_2$. We sketch how we can simulate $(\mathsf{ct}_{\mathbf{x},\mathbf{z}}, \mathsf{sk}_f)$ given $\mathbf{x}, f, f(\mathbf{x})^\top\mathbf{z}$; the proof extends readily to the many-key setting. The idea is to program

$$\tilde{\mathbf{w}} = \mathbf{z} + s\mathbf{w},\ \tilde{v} = s(\mathbf{u}^\top\mathbf{x}+v)$$

In addition, using (7), we can rewrite $(\mathsf{ct}_{\mathbf{x},\mathbf{z}}, \mathsf{sk}_f)$ as

$$\mathsf{ct}_{\mathbf{x},\mathbf{z}} = \left([s]_1, [\tilde{\mathbf{w}}]_1, [\tilde{v}]_1\right) \in \mathbb{G}_1^{n'+2}$$
$$\mathsf{sk}_f = \left([\underline{\mathbf{t}}+s^{-1}(\tilde{\mathbf{w}}-\mathbf{z})]_2, [\hat{\mathbf{u}}^\top]_2, [\mathbf{t}^\top(\mathbf{L}_1(\mathbf{x}\otimes\mathbf{I}_m)+\mathbf{L}_0)-\hat{\mathbf{u}}^\top\cdot(\mathbf{x}\otimes\mathbf{I}_m)+s^{-1}\tilde{v}\mathbf{r}^\top]_2, [\mathbf{r}]_2\right)$$

where $\hat{\mathbf{u}}^\top := \mathbf{t}^\top\mathbf{L}_1+\mathbf{u}^\top(\mathbf{I}_n\otimes\mathbf{r}^\top)$. Under the DDH assumption in $\mathbb{G}_2$, we know that[5]

$$[\mathbf{u}^\top(\mathbf{I}_n\otimes\mathbf{r}^\top)]_2, [\mathbf{r}^\top]_2, \mathbf{u}\leftarrow\mathbb{Z}_p^n, \mathbf{r}\leftarrow\mathbb{Z}_p^m$$

is pseudorandom, which means that $[\hat{\mathbf{u}}^\top]_2, [\mathbf{r}^\top]_2$ is pseudorandom.

We can therefore simulate $(\mathsf{ct}_{\mathbf{x},\mathbf{z}}, \mathsf{sk}_f)$ as follows: on input $\mu = f(\mathbf{x})^\top\mathbf{z}$,

1. run the simulator for partial garbling on input $f, \mathbf{x}, \mu$ to obtain $(\mathbf{p}_1^\top, \mathbf{p}_2^\top)$;
2. sample $s\leftarrow\mathbb{Z}_p, \tilde{\mathbf{w}}\leftarrow\mathbb{Z}_p^{n'}, \tilde{v}\leftarrow\mathbb{Z}_p, \hat{\mathbf{u}}\leftarrow\mathbb{Z}_p^{mn}$;
3. output

$$\mathsf{ct}_{\mathbf{x},\mathbf{z}} = \left([s]_1, [\tilde{\mathbf{w}}]_1, [\tilde{v}]_1\right) \in \mathbb{G}_1^{n'+2}$$
$$\mathsf{sk}_f = \left([-\mathbf{p}_1+s^{-1}\tilde{\mathbf{w}}]_2, [\hat{\mathbf{u}}^\top]_2, [\mathbf{p}_2^\top-\hat{\mathbf{u}}^\top\cdot(\mathbf{x}\otimes\mathbf{I}_m)+s^{-1}\tilde{v}\mathbf{r}^\top]_2, [\mathbf{r}]_2\right)$$

Looking ahead, we note that the above analysis extends to the $k$-Lin assumption, at the cost of blowing up the width of $\mathbf{u}, v, \mathbf{r}^\top$ by a factor of $k$. In the analysis, we use the fact that under $k$-Lin over $\mathbb{G}_2$, $([\mathbf{u}^\top(\mathbf{I}_n\otimes\mathbf{R})]_2, [\mathbf{R}]_2)$ is pseudorandom where $\mathbf{u}\leftarrow\mathbb{Z}_p^{kn}, \mathbf{R}\leftarrow\mathbb{Z}_p^{k\times m}$.

[5] Recall that if we write $\mathbf{u}=(u_1,\ldots,u_n)$, then $\mathbf{u}^\top(\mathbf{I}_n\otimes\mathbf{r}^\top)=(u_1\mathbf{r}^\top,\ldots,u_n\mathbf{r}^\top)$.

**The compiler.** To obtain a public-key scheme secure under the $k$-Lin assumption, we perform the following substitutions to (6), following [15,40]:

$$s \mapsto \mathbf{s}^\top \mathbf{A}^\top \in \mathbb{Z}_p^{1\times(k+1)},\ \mathbf{r}^\top \mapsto \mathbf{R} \in \mathbb{Z}_p^{k\times m},\ \mathbf{t}^\top \mapsto \mathbf{T} \in \mathbb{Z}_p^{(k+1)\times(m+n'-1)}$$

$$\mathbf{w}^\top \mapsto \mathbf{W} \in \mathbb{Z}_p^{(k+1)\times n'},\ \mathbf{u}^\top \mapsto \mathbf{U} \in \mathbb{Z}_p^{(k+1)\times kn},\ v \mapsto \mathbf{V} \in \mathbb{Z}_p^{(k+1)\times k}$$

That is, we blow up the height of $\mathbf{w}^\top, \mathbf{u}^\top, v, \mathbf{t}^\top$ by a factor of $k+1$, and the width of $\mathbf{u}^\top, v, \mathbf{r}$ by a factor of $k$. The proof of security follows the high-level strategy in [40]:

- We first switch $[\mathbf{s}^\top \mathbf{A}^\top]_1$ in the ciphertext with a random $[\mathbf{c}^\top]_1$.
- We decompose $\mathsf{sk}_f$ into two parts, $\mathbf{A}^\top \mathsf{sk}_f, \mathbf{c}^\top \mathsf{sk}_f$, corresponding to component-wise multiplication by $\mathbf{A}^\top, \mathbf{c}^\top$ respectively, using the fact that $(\mathbf{A}|\mathbf{c})$ forms a full-rank basis.
- We simulate $\mathbf{A}^\top \mathsf{sk}_f$ using $(\mathsf{mpk}, f)$, and simulate the ciphertext and $\mathbf{c}^\top \mathsf{sk}_f$ as in the secret-key setting we just described.

We refer the reader to Sect. 6 to see how the construction can be extended to handle the "extended" functionality in (3); an overview is given at the beginning of that section.

### 2.2 Unbounded-Slot Scheme

We refer the reader to Sect. 1.2 for a high-level overview of the unbounded-slot scheme, and proceed directly to describe the construction and the security proof.

**The construction.** We run two copies of the one-slot scheme, which we denote by $(\mathsf{Enc}_b, \mathsf{KeyGen}_b) = (\mathsf{Enc}(\mathsf{mpk}_b, \cdot), \mathsf{KeyGen}(\mathsf{msk}_b, \cdot))$ for $b = 1, 2$. We denote the corresponding simulators by $(\mathsf{Enc}_b^*, \mathsf{KeyGen}_b^*)$. Informally, we have

$$(\mathsf{Enc}_b(\mathbf{x}, \mathbf{z}\|w), \mathsf{KeyGen}_b(f, [r]_2)) \approx_c (\mathsf{Enc}_b^*(\mathbf{x}), \mathsf{KeyGen}_b^*((f, [r]_2), [f(\mathbf{x})^\top \mathbf{z} + wr]_2))$$

Then, $\mathsf{Enc}, \mathsf{KeyGen}$ in the unbounded-slot scheme are given by

$$\mathsf{Enc}((\mathbf{x}_i, \mathbf{z}_i)_i) = \mathsf{Enc}_1(\mathbf{x}_1, \mathbf{z}_1\| - \textstyle\sum_{i\in[2,N]} w_i),\ \mathsf{Enc}_2(\mathbf{x}_2, \mathbf{z}_2\|w_2), \cdots, \mathsf{Enc}_2(\mathbf{x}_N, \mathbf{z}_N\|w_N)$$
$$\mathsf{KeyGen}(f) = \mathsf{KeyGen}_1(f, [r]_2), \mathsf{KeyGen}_2(f, [r]_2), [r]_2$$

$$
\begin{array}{ll}
 & \mathsf{Enc}_1(\mathbf{x}_1, \mathbf{z}_1 \| -w_2 - w_3), \quad \mathsf{Enc}_2(\mathbf{x}_2, \mathbf{z}_2 \| w_2), \quad \mathsf{Enc}_2(\mathbf{x}_3, \mathbf{z}_3 \| w_3) \\
 & \mathsf{KeyGen}_1(f, [r]_2), \mathsf{KeyGen}_2(f, [r]_2) \\
\overset{\text{SIM-1}}{\approx_c} & \mathsf{Enc}_1^*(\mathbf{x}_1), \quad \mathsf{Enc}_2(\mathbf{x}_2, \mathbf{z}_2 \| w_2), \quad \mathsf{Enc}_2(\mathbf{x}_3, \mathbf{z}_3 \| w_3) \\
 & \mathsf{KeyGen}_1^*((f, [r]_2), [f(\mathbf{x}_1)^\top \mathbf{z}_1 - w_2 r - w_3 r]_2), \mathsf{KeyGen}_2(f, [r]_2) \\
\overset{\text{SIM-2}}{\approx_c} & \mathsf{Enc}_1^*(\mathbf{x}_1), \quad \mathsf{Enc}_2^*(\mathbf{x}_2), \quad \mathsf{Enc}_2(\mathbf{x}_3, \mathbf{z}_3 \| w_3) \\
 & \mathsf{KeyGen}_1^*((f, [r]_2), [f(\mathbf{x}_1)^\top \mathbf{z}_1 - w_2 r - w_3 r]_2), \mathsf{KeyGen}_2^*((f, [r]_2), [f(\mathbf{x}_2)^\top \mathbf{z}_2 + w_2 r]_2) \\
\overset{\text{DDH}}{\approx_c} & \mathsf{Enc}_1^*(\mathbf{x}_1), \quad \mathsf{Enc}_2^*(\mathbf{x}_2), \quad \mathsf{Enc}_2(\mathbf{x}_3, \mathbf{z}_3 \| w_3) \\
 & \mathsf{KeyGen}_1^*((f, [r]_2), [f(\mathbf{x}_1)^\top \mathbf{z}_1 + f(\mathbf{x}_2)^\top \mathbf{z}_2 - w_2 r - w_3 r]_2), \mathsf{KeyGen}_2^*((f, [r]_2), [w_2 r]_2) \\
\overset{\text{SIM-2}}{\approx_c} & \mathsf{Enc}_1^*(\mathbf{x}_1), \quad \mathsf{Enc}_2(\mathbf{x}_2, \mathbf{0} \| w_2), \quad \mathsf{Enc}_2(\mathbf{x}_3, \mathbf{z}_3 \| w_3) \\
 & \mathsf{KeyGen}_1^*((f, [r]_2), [f(\mathbf{x}_1)^\top \mathbf{z}_1 + f(\mathbf{x}_2)^\top \mathbf{z}_2 - w_2 r - w_3 r]_2), \mathsf{KeyGen}_2(f, [r]_2) \\
\overset{\text{SIM-2}}{\approx_c} & \mathsf{Enc}_1^*(\mathbf{x}_1), \quad \mathsf{Enc}_2(\mathbf{x}_2, \mathbf{0} \| w_2), \quad \mathsf{Enc}_2^*(\mathbf{x}_3) \\
 & \mathsf{KeyGen}_1^*((f, [r]_2), [f(\mathbf{x}_1)^\top \mathbf{z}_1 + f(\mathbf{x}_2)^\top \mathbf{z}_2 - w_2 r - w_3 r]_2), \mathsf{KeyGen}_2^*((f, [r]_2), [f(\mathbf{x}_3)^\top \mathbf{z}_3 + w_3 r]_2) \\
\overset{\text{DDH}}{\approx_c} & \mathsf{Enc}_1^*(\mathbf{x}_1), \quad \mathsf{Enc}_2(\mathbf{x}_2, \mathbf{0} \| w_2), \quad \mathsf{Enc}_2^*(\mathbf{x}_3) \\
 & \mathsf{KeyGen}_1^*((f, [r]_2), [f(\mathbf{x}_1)^\top \mathbf{z}_1 + f(\mathbf{x}_2)^\top \mathbf{z}_2 + f(\mathbf{x}_3)^\top \mathbf{z}_3 - w_2 r - w_3 r]_2), \mathsf{KeyGen}_2^*((f, [r]_2), [w_3 r]_2) \\
\overset{\text{SIM-2}}{\approx_c} & \mathsf{Enc}_1^*(\mathbf{x}_1), \quad \mathsf{Enc}_2(\mathbf{x}_2, \mathbf{0} \| w_2), \quad \mathsf{Enc}_2(\mathbf{x}_3, \mathbf{0} \| w_3) \\
 & \mathsf{KeyGen}_1^*((f, [r]_2), [f(\mathbf{x}_1)^\top \mathbf{z}_1 + f(\mathbf{x}_2)^\top \mathbf{z}_2 + f(\mathbf{x}_3)^\top \mathbf{z}_3 - w_2 r - w_3 r]_2), \mathsf{KeyGen}_2(f, [r]_2)
\end{array}
$$

**Fig. 3.** Summary of game sequence for $N = 3$. In the figure, $\overset{\text{SIM-}b}{\approx_c}$ indicates that this step uses the simulate-based semi-adaptive security of $(\mathsf{Enc}_b, \mathsf{KeyGen}_b)$.

The final simulator is given by:

$$
\begin{aligned}
\mathsf{Enc}^*((\mathbf{x}_i)_i) &= \mathsf{Enc}_1^*(\mathbf{x}_1), \mathsf{Enc}_2(\mathbf{x}_2, \mathbf{0} \| w_2), \cdots, \mathsf{Enc}_2(\mathbf{x}_N, \mathbf{0} \| w_N) \\
\mathsf{KeyGen}^*(f, \mu) &= \mathsf{KeyGen}_1^*((f, [r]_2), [\mu - \textstyle\sum_{i \in [2,N]} w_i r]_2), \mathsf{KeyGen}_2(f, [r]_2)
\end{aligned}
$$

As a sanity check, observe that decrypting $\mathsf{Enc}^*((\mathbf{x}_i)_i)$ using $\mathsf{KeyGen}^*(f, \sum_i f(\mathbf{x}_i)^\top \mathbf{z}_i)$ returns $\sum_i f(\mathbf{x}_i)^\top \mathbf{z}_i$.

**Proof overview.** For simplicity, we focus on the setting $N = 3$ with one secret key query in Fig. 3 where in $\overset{\text{DDH}}{\approx_c}$, we use pseudorandomness of $([w_1 r]_2, [r]_2)$ and $([w_2 r]_2, [r]_2)$ respectively; in $\overset{\text{SIM-1}}{\approx_c}$ and $\overset{\text{SIM-2}}{\approx_c}$, we use simulation-based semi-adaptive security of $(\mathsf{Enc}_1, \mathsf{KeyGen}_1)$ and $(\mathsf{Enc}_2, \mathsf{KeyGen}_2)$, respectively.

In the setting for general $N$ and $Q$ secret key queries,

- we will invoke simulation-based security of $(\mathsf{Enc}_1, \mathsf{KeyGen}_1)$ once, and that of $(\mathsf{Enc}_2, \mathsf{KeyGen}_2)$ for $2(N-1)$ times, while using the fact that both of these schemes are also secure against $Q$ secret key queries;
- in $\overset{\text{DDH}}{\approx_c}$, we will rely on pseudorandomness of $\{[w_i r_j]_2, [r_j]_2)\}_{j \in [Q]}$ for $i \in [2, N]$.

## 3 Preliminaries

**Notations.** We denote by $s \leftarrow S$ the fact that $s$ is picked uniformly at random from a finite set $S$. We use $\approx_s$ to denote two distributions being statistically

indistinguishable, and $\approx_c$ to denote two distributions being computationally indistinguishable. We use lower case boldface to denote *column* vectors and upper case boldcase to denote matrices. We use $\mathbf{e}_i$ to denote the $i$'th elementary column vector (with 1 at the $i$'th position and 0 elsewhere, and the total length of the vector specified by the context). For any positive integer $N$, we use $[N]$ to denote $\{1, 2, \ldots, N\}$ and $[2, N]$ to denote $\{2, \ldots, N\}$.

The tensor product (Kronecker product) for matrices $\mathbf{A} = (a_{i,j}) \in \mathbb{Z}^{\ell \times m}$, $\mathbf{B} \in \mathbb{Z}^{n \times p}$ is defined as

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{1,1}\mathbf{B}, \ldots, a_{1,m}\mathbf{B} \\ \ldots, \quad \ldots, \quad \ldots \\ a_{\ell,1}\mathbf{B}, \ldots, a_{\ell,m}\mathbf{B} \end{bmatrix} \in \mathbb{Z}^{\ell n \times mp}. \tag{8}$$

**Arithmetic Branching Programs.** A branching program is defined by a directed acyclic graph $(V, E)$, two special vertices $v_0, v_1 \in V$ and a labeling function $\phi$. An arithmetic branching program (ABP), where $p$ is a prime, computes a function $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$. Here, $\phi$ assigns to each edge in $E$ an affine function in some input variable or a constant, and $f(x)$ is the sum over all $v_0 - v_1$ paths of the product of all the values along the path. We refer to $|V| + |E|$ as the size of $f$. The definition extends in a coordinate-wise manner to functions $f : \mathbb{Z}_p^n \to \mathbb{Z}_p^{n'}$. Henceforth, we use $\mathcal{F}_{\mathsf{ABP},n,n'}$ to denote the class of ABP $f : \mathbb{Z}_p^n \to \mathbb{Z}_p^{n'}$.

We note that there is a linear-time algorithm that converts any boolean formula, boolean branching program or arithmetic formula to an arithmetic branching program with a constant blow-up in the representation size. Thus, ABPs can be viewed as a stronger computational model than all of the above. Recall also that branching programs and boolean formulas correspond to the complexity classes **LOGSPACE** and **NC1** respectively.

### 3.1 Prime-Order Bilinear Groups

A generator $\mathcal{G}$ takes as input a security parameter $1^\lambda$ and outputs a description $\mathbb{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, where $p$ is a prime of $\Theta(\lambda)$ bits, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of order $p$, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate bilinear map. We require that the group operations in $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ and the bilinear map $e$ are computable in deterministic polynomial time in $\lambda$. Let $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and $g_T = e(g_1, g_2) \in \mathbb{G}_T$ be the respective generators. We employ the *implicit representation* of group elements: for a matrix $\mathbf{M}$ over $\mathbb{Z}_p$, we define $[\mathbf{M}]_1 := g_1^{\mathbf{M}}, [\mathbf{M}]_2 := g_2^{\mathbf{M}}, [\mathbf{M}]_T := g_T^{\mathbf{M}}$, where exponentiation is carried out component-wise. Also, given $[\mathbf{A}]_1, [\mathbf{B}]_2$, we let $e([\mathbf{A}]_1, [\mathbf{B}]_2) = [\mathbf{AB}]_T$. We recall the matrix Diffie-Hellman (MDDH) assumption on $\mathbb{G}_1$ [20]:

**Assumption 1** ($\mathrm{MDDH}^d_{k,\ell}$ **Assumption**). *Let $k, \ell, d \in \mathbb{N}$. We say that the $\mathrm{MDDH}^d_{k,\ell}$ assumption holds if for all PPT adversaries $\mathcal{A}$, the following advantage function is negligible in $\lambda$.*

$$\mathsf{Adv}^{\mathrm{MDDH}^d_{k,\ell}}_{\mathcal{A}}(\lambda) := \left| \Pr[\mathcal{A}(\mathbb{G}, [\mathbf{M}]_1, \boxed{[\mathbf{MS}]_1}) = 1] - \Pr[\mathcal{A}(\mathbb{G}, [\mathbf{M}]_1, \boxed{[\mathbf{U}]_1}) = 1] \right|$$

*where $\mathbb{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$, $\mathbf{M} \leftarrow \mathbb{Z}_p^{\ell \times k}$, $\mathbf{S} \leftarrow \mathbb{Z}_p^{k \times d}$ and $\mathbf{U} \leftarrow \mathbb{Z}_p^{\ell \times d}$.*

The MDDH assumption on $\mathbb{G}_2$ can be defined in an analogous way. Escala *et al.* [20] showed that

$$k\text{-Lin} \Rightarrow \text{MDDH}^1_{k,k+1} \Rightarrow \text{MDDH}^d_{k,\ell}\ \forall\, k, d \geq 1, \ell > k$$

with a tight security reduction. (In the setting where $\ell \leq k$, the $\text{MDDH}^d_{k,\ell}$ assumption holds unconditionally.)

We state the following lemma implied by $\text{MDDH}^1_{k,Q}$ without proof.

**Lemma 1.** *For all $Q \in \mathbb{N}$ and $\mu_1, \ldots, \mu_Q \in \mathbb{Z}_p$, we have*

$$\begin{aligned} &\left\{ \quad [-\mathbf{w}^\top \mathbf{r}_j]_2,\ [\boxed{\mu_j} + \mathbf{w}^\top \mathbf{r}_j]_2,\ [\mathbf{r}_j]_2 \right\}_{j\in[Q]} \\ \approx_c &\left\{ [\mu_j - \mathbf{w}^\top \mathbf{r}_j]_2,\quad [\mathbf{w}^\top \mathbf{r}_j]_2,\ [\mathbf{r}_j]_2 \right\}_{j\in[Q]} \end{aligned}$$

*where $\mathbf{w}, \mathbf{r}_j \leftarrow \mathbb{Z}_p^k$ for all $j \in [Q]$. Concretely, the distinguishing advantage is bounded by $2 \cdot \mathsf{Adv}_{\mathcal{B}}^{\text{MDDH}^1_{k,Q}}(\lambda)$.*

## 4 Definitions and Tools

In this section, we formalize functional encryption for attribute-weighted sums, using the framework of partially-hiding functional encryption [13,24,40].

### 4.1 FE for Attribute-Weighted Sums

**Syntax.** An *unbounded-slot FE for attribute-weighted sums* consists of four algorithms:

$\mathsf{Setup}(1^\lambda, 1^n, 1^{n'})$ : The setup algorithm gets as input the security parameter $1^\lambda$ and function parameters $1^n, 1^{n'}$. It outputs the master public key $\mathsf{mpk}$ and the master secret key $\mathsf{msk}$.

$\mathsf{Enc}(\mathsf{mpk}, (\mathbf{x}_i, \mathbf{z}_i)_{i\in[N]})$ : The encryption algorithm gets as input $\mathsf{mpk}$ and message $(\mathbf{x}_i, \mathbf{z}_i)_{i\in[N]} \in (\mathbb{Z}_p^n \times \mathbb{Z}_p^{n'})^\star$. It outputs a ciphertext $\mathsf{ct}_{(\mathbf{x}_i,\mathbf{z}_i)}$ with $(\mathbf{x}_i)$ being public.

$\mathsf{KeyGen}(\mathsf{msk}, f)$ : The key generation algorithm gets as input $\mathsf{msk}$ and a function $f \in \mathcal{F}_{\mathsf{ABP},n,n'}$. It outputs a secret key $\mathsf{sk}_f$ with $f$ being public.

$\mathsf{Dec}((\mathsf{sk}_f, f), (\mathsf{ct}_{(\mathbf{x}_i,\mathbf{z}_i)}, (\mathbf{x}_i)_{i\in[N]}))$ : The decryption algorithm gets as input $\mathsf{sk}_f$ and $\mathsf{ct}_{(\mathbf{x}_i,\mathbf{z}_i)}$ along with $f$ and $(\mathbf{x}_i)_{i\in[N]}$. It outputs a value in $\mathbb{Z}_p$.

**Correctness.** For all $(\mathbf{x}_i, \mathbf{z}_i)_{i\in[N]} \in (\mathbb{Z}_p^n \times \mathbb{Z}_p^{n'})^\star$ and $f \in \mathcal{F}_{\mathsf{ABP},n,n'}$, we require

$$\Pr[\mathsf{Dec}((\mathsf{ct}_{(\mathbf{x}_i,\mathbf{z}_i)}, (\mathbf{x}_i)_{i\in[N]}), (\mathsf{sk}_f, f)) = \sum_{i\in[N]} f(\mathbf{x}_i)^\top \mathbf{z}_i] = 1$$

where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^{n'})$, $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$ and $\mathsf{ct}_{(\mathbf{x}_i,\mathbf{z}_i)} \leftarrow \mathsf{Enc}(\mathsf{mpk}, (\mathbf{x}_i, \mathbf{z}_i)_{i\in[N]})$.

*Remark 1 (Relaxation of correctness.).* Our scheme only achieves a relaxation of correctness where the decryption algorithm takes an additional bound $1^B$ (and runs in time polynomial in $B$) and outputs $\sum_{i\in[N]} f(\mathbf{x}_i)^\top \mathbf{z}_i$ if the value is bounded by $B$. This limitation is also present in prior works on (IP)FE from DDH and bilinear groups [1,3,6,10,33], due to the reliance on brute-force discrete log to recover the answer "from the exponent". We stress that the relaxation only refers to functionality and does not affect security.

**Security definition.** We consider semi-adaptive [16] (strengthening of selective), simulation-based security, which stipulates that there exists a randomized simulator $(\mathsf{Setup}^*, \mathsf{Enc}^*, \mathsf{KeyGen}^*)$ such that for every efficient stateful adversary $\mathcal{A}$,

$$\left[\begin{array}{l} 1^N \leftarrow \mathcal{A}(1^\lambda); \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^{n'}); \\ (\mathbf{x}_i^*, \mathbf{z}_i^*)_{i\in[N]} \leftarrow \mathcal{A}(\mathsf{mpk}); \\ \mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, (\mathbf{x}_i^*, \mathbf{z}_i^*)_{i\in[N]}); \\ \text{output } \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{mpk}, \mathsf{ct}^*) \end{array}\right] \approx_c \left[\begin{array}{l} 1^N \leftarrow \mathcal{A}(1^\lambda); \\ (\mathsf{mpk}, \mathsf{msk}^*) \leftarrow \mathsf{Setup}^*(1^\lambda, 1^n, 1^{n'}, 1^N); \\ (\mathbf{x}_i^*, \mathbf{z}_i^*)_{i\in[N]} \leftarrow \mathcal{A}(\mathsf{mpk}); \\ \mathsf{ct}^* \leftarrow \mathsf{Enc}^*(\mathsf{msk}^*, (\mathbf{x}_i^*)_{i\in[N]}); \\ \text{output } \mathcal{A}^{\mathsf{KeyGen}^*(\mathsf{msk}^*, (\mathbf{x}_i^*)_{i\in[N]}, \cdot, \cdot)}(\mathsf{mpk}, \mathsf{ct}^*) \end{array}\right]$$

such that whenever $\mathcal{A}$ makes a query $f$ to $\mathsf{KeyGen}$, the simulator $\mathsf{KeyGen}^*$ gets $f$ along with $\sum_{i\in[N]} f(\mathbf{x}_i^*)^\top \mathbf{z}_i^*$. We use $\mathsf{Adv}^{\mathrm{FE}}_{\mathcal{A}}(\lambda)$ to denote the advantage in distinguishing the real and ideal games.

**One-slot scheme.** A *one-slot* scheme is the same thing, except we always have $N = 1$ for both correctness and security.

### 4.2 Partial Garbling Scheme

The partial garbling scheme [27,40] for $f(\mathbf{x})^\top \mathbf{z}$ with $f \in \mathcal{F}_{\mathsf{ABP},n,n'}$ is a randomized algorithm that on input $f$ outputs an affine function in $\mathbf{x}, \mathbf{z}$ of the form:

$$\mathbf{p}^\top_{f,\mathbf{x},\mathbf{z}} = \left(\mathbf{z}^\top - \underline{\mathbf{t}}^\top, \mathbf{t}^\top(\mathbf{L}_1(\mathbf{x}\otimes\mathbf{I}_m) + \mathbf{L}_0)\right)$$

where $\mathbf{L}_0 \in \mathbb{Z}_p^{(m+n'-1)\times mn}, \mathbf{L}_1 \in \mathbb{Z}_p^{(m+n'-1)\times m}$ depends only on $f$; $\mathbf{t} \leftarrow \mathbb{Z}_p^{m+n'-1}$ is the random coin and $\underline{\mathbf{t}}$ consists of the last $n'$ entries in $\mathbf{t}$, such that given $(\mathbf{p}^\top_{f,\mathbf{x},\mathbf{z}}, f, \mathbf{x})$, we can recover $f(\mathbf{x})^\top \mathbf{z}$, while learning nothing else about $\mathbf{z}$.

**Lemma 2 (partial garbling [27,40]).** *There exists four efficient algorithms* $(\mathsf{lgen}, \mathsf{pgb}, \mathsf{rec}, \mathsf{pgb}^*)$ *with the following properties:*

- *syntax: on input* $f \in \mathcal{F}_{\mathsf{ABP},n,n'}$, $\mathsf{lgen}(f)$ *outputs* $\mathbf{L}_0 \in \mathbb{Z}_p^{(m+n'-1)\times mn}, \mathbf{L}_1 \in \mathbb{Z}_p^{(m+n'-1)\times m}$, *and*

$$\begin{aligned} \mathsf{pgb}(f, \mathbf{x}, \mathbf{z}; \mathbf{t}) &= \left(\mathbf{z}^\top - \underline{\mathbf{t}}^\top, \mathbf{t}^\top(\mathbf{L}_1(\mathbf{x}\otimes\mathbf{I}_m) + \mathbf{L}_0)\right) \\ \mathsf{pgb}^*(f, \mathbf{x}, \mu; \mathbf{t}) &= \left(\quad -\underline{\mathbf{t}}^\top, \mathbf{t}^\top(\mathbf{L}_1(\mathbf{x}\otimes\mathbf{I}_m) + \mathbf{L}_0) + \mu\cdot\mathbf{e}_1^\top\right) \end{aligned}$$

  *where* $\mathbf{t} \in \mathbb{Z}_p^{m+n'-1}$ *and* $\underline{\mathbf{t}}$ *consists of the last* $n'$ *entries in* $\mathbf{t}$ *and* $m$ *are linear in the size of* $f$.

– *reconstruction:* $\mathsf{rec}(f, \mathbf{x})$ *outputs* $\mathbf{d}_{f,\mathbf{x}} \in \mathbb{Z}_p^{n'+m}$ *such that for all* $f, \mathbf{x}, \mathbf{z}, \mathbf{t}$*, we have* $\mathbf{p}_{f,\mathbf{x},\mathbf{z}}^\top \mathbf{d}_{f,\mathbf{x}} = f(\mathbf{x})^\top \mathbf{z}$ *where* $\mathbf{p}_{f,\mathbf{x},\mathbf{z}}^\top = \mathsf{pgb}(f, \mathbf{x}, \mathbf{z}; \mathbf{t})$.
– *privacy: for all* $f, \mathbf{x}, \mathbf{z}$*,* $\mathsf{pgb}(f, \mathbf{x}, \mathbf{z}; \mathbf{t}) \approx_s \mathsf{pgb}^*(f, \mathbf{x}, f(\mathbf{x})^\top \mathbf{z}; \mathbf{t})$ *where the randomness is over* $\mathbf{t} \leftarrow \mathbb{Z}_p^{m+n'-1}$.

**Extension.** We will also rely on an extra property of the above construction to handle shifts by $\delta \in \mathbb{Z}_p$, namely that, given

$$\mathbf{p}_{f,\mathbf{x},\mathbf{z},\boxed{\delta}}^\top = \left(\mathbf{z}^\top - \underline{\mathbf{t}}^\top, \mathbf{t}^\top(\mathbf{L}_1(\mathbf{x} \otimes \mathbf{I}_m) + \mathbf{L}_0) + \boxed{\delta \cdot \mathbf{e}_1^\top}\right)$$

together with $(f, \mathbf{x})$, we can recover $f(\mathbf{x})^\top \mathbf{z} + \delta$, while learning nothing else about $\mathbf{z}, \delta$. That is, for all $f, \mathbf{x}, \mathbf{z}$ and $\delta \in \mathbb{Z}_p$:

– reconstruction: $(\mathsf{pgb}(f, \mathbf{x}, \mathbf{z}; \mathbf{t}) + (\mathbf{0}, \boxed{\delta} \cdot \mathbf{e}_1^\top))\mathbf{d}_{f,\mathbf{x}} = f(\mathbf{x})^\top \mathbf{z} + \boxed{\delta}$;
– privacy: $\mathsf{pgb}(f, \mathbf{x}, \mathbf{z}; \mathbf{t}) + (\mathbf{0}, \boxed{\delta} \cdot \mathbf{e}_1^\top) \approx_s \mathsf{pgb}^*(f, \mathbf{x}, f(\mathbf{x})^\top \mathbf{z} + \boxed{\delta}; \mathbf{t})$ where the randomness is over $\mathbf{t} \leftarrow \mathbb{Z}_p^{m+n'-1}$.

See the full paper for more detail about Lemma 2 and the extension.

## 5 $\Pi_{\mathsf{one}}$: One-Slot Scheme

In this section, we present our one-slot FE scheme for attribute-weighted sums. This scheme achieves simulation-based semi-adaptive security under $k$-Linear assumptions.

### 5.1 Construction

Our one-slot FE scheme $\Pi_{\mathsf{one}}$ in prime-order bilinear group is described as follows.

– $\mathsf{Setup}(1^\lambda, 1^n, 1^{n'})$: Run $\mathbb{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$. Sample

$$\mathbf{A} \leftarrow \mathbb{Z}_p^{(k+1)\times k} \quad \text{and} \quad \mathbf{W} \leftarrow \mathbb{Z}_p^{(k+1)\times n'}, \mathbf{U} \leftarrow \mathbb{Z}_p^{(k+1)\times kn}, \mathbf{V} \leftarrow \mathbb{Z}_p^{(k+1)\times k}$$

and output

$$\mathsf{mpk} = \left(\mathbb{G}, [\mathbf{A}^\top]_1, [\mathbf{A}^\top \mathbf{W}]_1, [\mathbf{A}^\top \mathbf{U}]_1, [\mathbf{A}^\top \mathbf{V}]_1\right) \quad \text{and} \quad \mathsf{msk} = \left(\mathbf{W}, \mathbf{U}, \mathbf{V}\right).$$

– $\mathsf{Enc}(\mathsf{mpk}, (\mathbf{x}, \mathbf{z}))$: Sample $\mathbf{s} \leftarrow \mathbb{Z}_p^k$ and output

$$\mathsf{ct}_{\mathbf{x},\mathbf{z}} = \left([\mathbf{s}^\top \mathbf{A}^\top]_1, [\mathbf{z}^\top + \mathbf{s}^\top \mathbf{A}^\top \mathbf{W}]_1, [\mathbf{s}^\top \mathbf{A}^\top \mathbf{U}(\mathbf{x} \otimes \mathbf{I}_k) + \mathbf{s}^\top \mathbf{A}^\top \mathbf{V}]_1\right) \quad \text{and} \quad \mathbf{x}.$$

– $\mathsf{KeyGen}(\mathsf{msk}, f)$: Run $(\mathbf{L}_1, \mathbf{L}_0) \leftarrow \mathsf{lgen}(f)$ where $\mathbf{L}_1 \in \mathbb{Z}_p^{(m+n'-1)\times mn}, \mathbf{L}_0 \in \mathbb{Z}_p^{(m+n'-1)\times m}$ (cf. Sect. 4.2). Sample $\mathbf{T} \leftarrow \mathbb{Z}_p^{(k+1)\times(m+n'-1)}$ and $\mathbf{R} \leftarrow \mathbb{Z}_p^{k\times m}$ and output

$$\mathsf{sk}_f = \left([\underline{\mathbf{T}} + \mathbf{W}]_2, [\mathbf{T}\mathbf{L}_1 + \mathbf{U}(\mathbf{I}_n \otimes \mathbf{R})]_2, [\mathbf{T}\mathbf{L}_0 + \mathbf{V}\mathbf{R}]_2, [\mathbf{R}]_2\right) \quad \text{and} \quad f$$

where $\underline{\mathbf{T}}$ refers to the matrix composed of the right most $n'$ columns of $\mathbf{T}$.

– $\mathsf{Dec}((\mathsf{sk}_f, f), (\mathsf{ct}_{\mathbf{x},\mathbf{z}}, \mathbf{x}))$: On input key:

$$\mathsf{sk}_f = \left( [\mathbf{K}_1]_2, [\mathbf{K}_2]_2, [\mathbf{K}_3]_2, [\mathbf{R}]_2 \right) \quad \text{and} \quad f$$

and ciphertext:

$$\mathsf{ct}_{\mathbf{x},\mathbf{z}} = \left( [\mathbf{c}_0^\top]_1, [\mathbf{c}_1^\top]_1, [\mathbf{c}_2^\top]_1 \right) \quad \text{and} \quad \mathbf{x}$$

the decryption works as follows:

1. compute
$$[\mathbf{p}_1^\top]_T = e([\mathbf{c}_1^\top]_1, [\mathbf{I}_{n'}]_2) \cdot e([\mathbf{c}_0^\top]_1, [-\mathbf{K}_1]_2) \tag{9}$$
2. compute
$$[\mathbf{p}_2^\top]_T = e([\mathbf{c}_0^\top]_1, [\mathbf{K}_2(\mathbf{x} \otimes \mathbf{I}_m) + \mathbf{K}_3]_2) \cdot e([-\mathbf{c}_2^\top]_1, [\mathbf{R}]_2) \tag{10}$$
3. run $\mathbf{d}_{f,\mathbf{x}} \leftarrow \mathsf{rec}(f, \mathbf{x})$ (cf. Sect. 4.2), compute
$$[D]_T = [(\mathbf{p}_1^\top, \mathbf{p}_2^\top)\mathbf{d}_{f,\mathbf{x}}]_T \tag{11}$$
and use brute-force discrete log to recover $D$ as the output.

**Correctness.** For $\mathsf{ct}_{\mathbf{x},\mathbf{z}}$ and $\mathsf{sk}_f$, we have

$$\mathbf{p}_1^\top = \mathbf{z}^\top - \mathbf{s}^\top \mathbf{A}^\top \underline{\mathbf{T}} \tag{12}$$
$$\mathbf{p}_2^\top = \mathbf{s}^\top \mathbf{A}^\top \mathbf{T}\mathbf{L}_1(\mathbf{x} \otimes \mathbf{I}_m) + \mathbf{s}^\top \mathbf{A}^\top \mathbf{T}\mathbf{L}_0 \tag{13}$$
$$(\mathbf{p}_1^\top, \mathbf{p}_2^\top)\mathbf{d}_{f,\mathbf{x}} = f(\mathbf{x})^\top \mathbf{z} \tag{14}$$

Here (14) follows from the fact that

$$(\mathbf{p}_1^\top, \mathbf{p}_2^\top) = \mathsf{pgb}(f, \mathbf{x}, \mathbf{z}; (\mathbf{s}^\top \mathbf{A}^\top \mathbf{T})^\top) \quad \text{and} \quad \mathbf{d}_{f,\mathbf{x}} = \mathsf{rec}(f, \mathbf{x})$$

and reconstruction of the partial garbling in (9); the remaining two equalities follow from:

$$(12)\quad \mathbf{z}^\top - \mathbf{s}^\top \mathbf{A}^\top \underline{\mathbf{T}} = (\mathbf{z}^\top + \mathbf{s}^\top \mathbf{A}^\top \mathbf{W}) \cdot \mathbf{I}_{n'} - \mathbf{s}^\top \mathbf{A}^\top \cdot (\underline{\mathbf{T}} + \mathbf{W})$$
$$(13)\quad \mathbf{s}^\top \mathbf{A}^\top \mathbf{T}\mathbf{L}_1(\mathbf{x} \otimes \mathbf{I}_m) + \mathbf{s}^\top \mathbf{A}^\top \mathbf{T}\mathbf{L}_0 = \mathbf{s}^\top \mathbf{A}^\top \cdot ((\mathbf{T}\mathbf{L}_1 + \mathbf{U}(\mathbf{I}_n \otimes \mathbf{R}))(\mathbf{x} \otimes \mathbf{I}_m) + (\mathbf{T}\mathbf{L}_0 + \mathbf{V}\mathbf{R})) - (\mathbf{s}^\top \mathbf{A}^\top \mathbf{U}(\mathbf{x} \otimes \mathbf{I}_k) + \mathbf{s}^\top \mathbf{A}^\top \mathbf{V}) \cdot \mathbf{R}$$

in which we use the equality $(\mathbf{I}_n \otimes \mathbf{R})(\mathbf{x} \otimes \mathbf{I}_m) = (\mathbf{x} \otimes \mathbf{I}_k)\mathbf{R}$. This readily proves the correctness.

*Remark 2 (Comparison with W17 [40]).* The ciphertext in [40] contains a term of the form

$$[\mathbf{x}^\top \otimes \mathbf{s}^\top \mathbf{A}^\top \mathbf{V} + \mathbf{s}^\top \mathbf{A}^\top \mathbf{U}]_1 \in \mathbb{G}_1^{kn} \quad \text{in the place of} \quad [\mathbf{s}^\top \mathbf{A}^\top \mathbf{U}(\mathbf{x} \otimes \mathbf{I}_k) + \mathbf{s}^\top \mathbf{A}^\top \mathbf{V}]_1 \in \mathbb{G}_1^k$$

where $\mathbf{U} \leftarrow \mathbb{Z}_p^{(k+1)\times kn}, \mathbf{V} \leftarrow \mathbb{Z}_p^{(k+1)\times k}$. The secret key sizes in both our schemes and that in [40] are $O(mn + n')$. In our scheme, the multiplicative factor of $n$ comes at the cost of a smaller ciphertext. In [40], the multiplicative factor of $n$ comes from a locality requirement that each column of $\mathbf{L}_1(\mathbf{x} \otimes \mathbf{I}_m) + \mathbf{L}_0$ depends on a single entry of $\mathbf{x}$, which can be achieved generically at the cost of a blow-up of $n$. We remove the locality requirement in our scheme.

**Security.** We have the following theorem with the proof shown in the subsequent subsection.

**Theorem 1.** *Our one-slot scheme* $\Pi_{\mathsf{one}}$ *for attribute-weighted sums described in this section achieves simulation-based semi-adaptive security under the MDDH assumption in* $\mathbb{G}_1$ *and in* $\mathbb{G}_2$*.*

### 5.2 Simulator

We start by describing the simulator.

– $\mathsf{Setup}^*(1^\lambda, 1^n, 1^{n'})$: Run $\mathbb{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$. Sample

$$\begin{array}{lll} \mathbf{A} \leftarrow \mathbb{Z}_p^{(k+1)\times k} & \text{and} & \mathbf{W} \leftarrow \mathbb{Z}_p^{(k+1)\times n'}, \mathbf{U} \leftarrow \mathbb{Z}_p^{(k+1)\times kn}, \mathbf{V} \leftarrow \mathbb{Z}_p^{(k+1)\times k} \\ \mathbf{c} \leftarrow \mathbb{Z}_p^{k+1} & & \widetilde{\mathbf{w}} \leftarrow \mathbb{Z}_p^{n'}, \qquad\qquad\qquad\quad \widetilde{\mathbf{v}} \leftarrow \mathbb{Z}_p^{k} \end{array}$$

and output

$$\begin{aligned} \mathsf{mpk} &= \big(\mathbb{G},\, [\mathbf{A}^\top]_1,\, [\mathbf{A}^\top\mathbf{W}]_1,\, [\mathbf{A}^\top\mathbf{U}]_1,\, [\mathbf{A}^\top\mathbf{V}]_1\big) \\ \mathsf{msk}^* &= \big(\mathbf{W},\, \mathbf{U},\, \mathbf{V},\, \widetilde{\mathbf{w}},\, \widetilde{\mathbf{v}},\, \mathbf{c}, \mathbf{C}^\perp, \mathbf{A}, \mathbf{a}^\perp\big) \end{aligned}$$

where $(\mathbf{A}|\mathbf{c})^\top(\mathbf{C}^\perp|\mathbf{a}^\perp) = \mathbf{I}_{k+1}$. Here we assume that $(\mathbf{A}|\mathbf{c})$ has full rank, which happens with probability $1 - 1/p$.

– $\mathsf{Enc}^*(\mathsf{msk}^*, \mathbf{x}^*)$: Output

$$\mathsf{ct}^* = \big([\mathbf{c}^\top]_1,\, [\widetilde{\mathbf{w}}^\top]_1,\, [\widetilde{\mathbf{v}}^\top]_1\big) \quad \text{and} \quad \mathbf{x}^*.$$

– $\mathsf{KeyGen}^*(\mathsf{msk}^*, \mathbf{x}^*, f, \mu \in \mathbb{Z}_p)$: Run

$$(\mathbf{L}_1, \mathbf{L}_0) \leftarrow \mathsf{lgen}(f) \quad \text{and} \quad ((\mathbf{p}_1^*)^\top, (\mathbf{p}_2^*)^\top) \leftarrow \mathsf{pgb}^*(f, \mathbf{x}^*, \mu).$$

Sample $\hat{\mathbf{u}} \leftarrow \mathbb{Z}_p^{nm}$, $\mathbf{T} \leftarrow \mathbb{Z}_p^{(k+1)\times(m+n'-1)}$ and $\mathbf{R} \leftarrow \mathbb{Z}_p^{k\times m}$ and output

$$\mathsf{sk}_f^* = \big(\mathbf{C}^\perp \cdot \mathsf{sk}_f^*[1] + \mathbf{a}^\perp \cdot \mathsf{sk}_f^*[2],\, [\mathbf{R}]_2\big) \quad \text{and} \quad f \tag{15}$$

where

$$\begin{aligned} \mathsf{sk}_f^*[1] &= \big([\mathbf{A}^\top\underline{\mathbf{T}} + \mathbf{A}^\top\mathbf{W}]_2,\, [\mathbf{A}^\top\mathbf{T}\mathbf{L}_1 + \mathbf{A}^\top\mathbf{U}(\mathbf{I}_n \otimes \mathbf{R})]_2,\, [\mathbf{A}^\top\mathbf{T}\mathbf{L}_0 + \mathbf{A}^\top\mathbf{V}\mathbf{R}]_2\big) \\ \mathsf{sk}_f^*[2] &= \big([-(\mathbf{p}_1^*)^\top + \widetilde{\mathbf{w}}^\top]_2,\, [\hat{\mathbf{u}}^\top]_2,\, [(\mathbf{p}_2^*)^\top - \hat{\mathbf{u}}^\top(\mathbf{x}^* \otimes \mathbf{I}_m) + \widetilde{\mathbf{v}}^\top\mathbf{R}]_2\big) \end{aligned}$$

Here $\underline{\mathbf{T}}$ refers to the matrix composed of the right most $n'$ columns of $\mathbf{T}$. That is,

$$\mathsf{sk}_f^* = \begin{pmatrix} [\mathbf{C}^\perp(\mathbf{A}^\top\underline{\mathbf{T}} + \mathbf{A}^\top\mathbf{W}) & +\mathbf{a}^\perp(-(\mathbf{p}_1^*)^\top + \widetilde{\mathbf{w}}^\top)]_2, \\ [\mathbf{C}^\perp(\mathbf{A}^\top\mathbf{T}\mathbf{L}_1 + \mathbf{A}^\top\mathbf{U}(\mathbf{I}_n \otimes \mathbf{R})) & +\mathbf{a}^\perp(\hat{\mathbf{u}}^\top)]_2 \\ [\mathbf{C}^\perp(\mathbf{A}^\top\mathbf{T}\mathbf{L}_0 + \mathbf{A}^\top\mathbf{V}\mathbf{R}) & +\mathbf{a}^\perp\big((\mathbf{p}_2^*)^\top - \hat{\mathbf{u}}^\top(\mathbf{x}^* \otimes \mathbf{I}_m) + \widetilde{\mathbf{v}}^\top\mathbf{R}\big)]_2 \end{pmatrix}, [\mathbf{R}]_2 \Bigg)$$

*Remark 3 (decryption checks).* As a sanity check, we check that an adversary cannot use the decryption algorithm to distinguish between the real and simulated output.

Observe that when we decrypt the simulated ciphertext $\mathsf{ct}^*_{\mathbf{x}^*} \leftarrow \mathsf{Enc}^*(\mathsf{msk}^*, \mathbf{x}^*)$ with the simulated secret key $\mathsf{sk}^*_f \leftarrow \mathsf{KeyGen}^*(\mathsf{msk}^*, \mathbf{x}^*, f, f(\mathbf{x}^*)^\top \mathbf{z}^*)$, the $\mathsf{sk}^*_f[1]$ part cancels out and leaves just the $\mathsf{sk}^*_f[2]$ part since $\mathbf{c}^\top \mathbf{C}^\perp = \mathbf{0}, \mathbf{c}^\top \mathbf{a}^\perp = 1$ and we end up with $((\mathbf{p}^*_1)^\top, (\mathbf{p}^*_2)^\top)\mathbf{d}_{f,\mathbf{x}^*} = f(\mathbf{x}^*)^\top \mathbf{z}^*$ where $((\mathbf{p}^*_1)^\top, (\mathbf{p}^*_2)^\top) \leftarrow \mathsf{pgb}^*(f, \mathbf{x}^*, f(\mathbf{x}^*)^\top \mathbf{z}^*)$.

Similarly, when we decrypt a normal ciphertext $\mathsf{ct}_{\mathbf{x},\mathbf{z}} \leftarrow \mathsf{Enc}(\mathsf{mpk}, (\mathbf{x}, \mathbf{z}))$ corresponding to any $(\mathbf{x}, \mathbf{z})$ with a simulated secret key, the $\mathsf{sk}^*_f[2]$ part cancels out and leaves just the $\mathsf{sk}^*_f[1]$ part since $\mathbf{A}^\top \mathbf{C}^\perp = \mathbf{I}, \mathbf{A}^\top \mathbf{a}^\perp = \mathbf{0}$. We end up with $(\mathbf{p}^\top_1, \mathbf{p}^\top_2)\mathbf{d}_{f,\mathbf{x}} = f(\mathbf{x})^\top \mathbf{z}$ where $(\mathbf{p}^\top_1, \mathbf{p}^\top_2) = \mathsf{pgb}(f, \mathbf{x}, \mathbf{z}; (\mathbf{s}^\top \mathbf{A}^\top \mathbf{T})^\top)$ as in the real $\mathsf{Dec}$ algorithm.

### 5.3 Proof

With our simulator, we prove the following theorem which implies Theorem 1.

**Theorem 2.** *For all $\mathcal{A}$, there exist $\mathcal{B}_1$ and $\mathcal{B}_2$ with $\mathsf{Time}(\mathcal{B}_1), \mathsf{Time}(\mathcal{B}_2) \approx \mathsf{Time}(\mathcal{A})$ such that*

$$\mathsf{Adv}^{\Pi_{\mathsf{one}}}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{MDDH}^1_{k,k+1}}_{\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathrm{MDDH}^n_{k,mQ}}_{\mathcal{B}_2}(\lambda) + 1/p$$

*where $n$ is length of public input $\mathbf{x}^*$ in the challenge, $m$ is the parameter depending on size of function $f$ and $Q$ is the number of key queries.*

Note that this yields a tight security reduction to the $k$-Lin assumption. Before we proceed to describe the game sequence and proof, we state the following lemma we will use.

**Lemma 3 (statistical lemma).** *For any full-rank $(\mathbf{A}|\mathbf{c}) \in \mathbb{Z}_p^{(k+1)\times k} \times \mathbb{Z}_p^{k+1}$, we have*

$$\left\{ \mathbf{A}^\top \mathbf{W}, \boxed{\mathbf{c}^\top \mathbf{W}} : \mathbf{W} \leftarrow \mathbb{Z}_p^{(k+1)\times k} \right\} \equiv \left\{ \mathbf{A}^\top \mathbf{W}, \boxed{\tilde{\mathbf{w}}^\top} : \mathbf{W} \leftarrow \mathbb{Z}_p^{(k+1)\times k}, \boxed{\tilde{\mathbf{w}} \leftarrow \mathbb{Z}_p^k} \right\}.$$

**Game sequence.** We use $(\mathbf{x}^*, \mathbf{z}^*)$ to denote the semi-adaptive challenge and for notational simplicity, assume that all key queries $f_j$ share the same parameter $m$. We prove Theorem 2 via a series of games.

$\mathsf{Game}_0$: Real game.

$\mathsf{Game}_1$: Identical to $\mathsf{Game}_0$ except that $\mathsf{ct}^*$ for $(\mathbf{x}^*, \mathbf{z}^*)$ is given by

$$\mathsf{ct}^* = \left( [\boxed{\mathbf{c}^\top}]_1, [(\mathbf{z}^*)^\top + \boxed{\mathbf{c}^\top}\mathbf{W}]_1, [\boxed{\mathbf{c}^\top}\mathbf{U}(\mathbf{x}^* \otimes \mathbf{I}_k) + \boxed{\mathbf{c}^\top}\mathbf{V}]_1 \right)$$

where $\mathbf{c} \leftarrow \mathbb{Z}_p^{k+1}$. We claim that $\mathsf{Game}_0 \approx_c \mathsf{Game}_1$. This follows from $\mathrm{MDDH}^1_{k,k+1}$ assumption:

$$[\mathbf{A}^\top]_1, [\mathbf{s}^\top \mathbf{A}^\top]_1 \approx_c [\mathbf{A}^\top]_1, \boxed{[\mathbf{c}^\top]_1}.$$

In the reduction, we sample $\mathbf{W}, \mathbf{U}, \mathbf{V}$ honestly and use them to simulate $\mathsf{mpk}$ and $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ along with $[\mathbf{A}^\top]_1$; the challenge ciphertext $\mathsf{ct}^*$ is generated using the challenge term given above.

$\underline{\mathsf{Game}_2}$: Identical to $\mathsf{Game}_1$ except that the $j$-th query $f_j$ to $\mathsf{KeyGen}$ $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ is answered by

$$\mathsf{sk}_{f_j} = \left( \mathbf{C}^\perp \cdot \mathsf{sk}_{f_j}[1] + \mathbf{a}^\perp \cdot \mathsf{sk}_{f_j}[2],\ [\mathbf{R}_j]_2 \right)$$

with

$$\mathsf{sk}_{f_j}[1] = \left( [\mathbf{A}^\top \underline{\mathbf{T}}_j + \mathbf{A}^\top \mathbf{W}]_2,\ [\mathbf{A}^\top \mathbf{T}_j \mathbf{L}_{1,j} + \mathbf{A}^\top \mathbf{U}(\mathbf{I}_n \otimes \mathbf{R}_j)]_2,\ [\mathbf{A}^\top \mathbf{T}_j \mathbf{L}_{0,j} + \mathbf{A}^\top \mathbf{V}\mathbf{R}_j]_2 \right)$$
$$\mathsf{sk}_{f_j}[2] = \left( [\mathbf{c}^\top \underline{\mathbf{T}}_j + \mathbf{c}^\top \mathbf{W}]_2,\ [\mathbf{c}^\top \mathbf{T}_j \mathbf{L}_{1,j} + \mathbf{c}^\top \mathbf{U}(\mathbf{I}_n \otimes \mathbf{R}_j)]_2,\ [\mathbf{c}^\top \mathbf{T}_j \mathbf{L}_{0,j} + \mathbf{c}^\top \mathbf{V}\mathbf{R}_j]_2 \right)$$

where $(\mathbf{L}_{1,j}, \mathbf{L}_{0,j}) \leftarrow \mathsf{Igen}(f_j)$, $\mathbf{T}_j \leftarrow \mathbb{Z}_p^{(k+1)\times(m+n'-1)}$, $\mathbf{R}_j \leftarrow \mathbb{Z}_p^{k\times m}$, $\mathbf{c}$ is the randomness in $\mathsf{ct}^*$ and $\mathbf{C}^\perp, \mathbf{a}^\perp$ are defined such that $(\mathbf{A}|\mathbf{c})^\top (\mathbf{C}^\perp | \mathbf{a}^\perp) = \mathbf{I}_{k+1}$ (cf. $\mathsf{Setup}^*$ in Sect. 5.2). By basic linear algebra, we have $\mathsf{Game}_1 = \mathsf{Game}_2$.

$\underline{\mathsf{Game}_3}$: Identical to $\mathsf{Game}_2$ except that we replace $\mathsf{Setup}, \mathsf{Enc}$ with $\mathsf{Setup}^*, \mathsf{Enc}^*$ where $\mathsf{ct}^*$ is given by

$$\mathsf{ct}^* = \left( [\mathbf{c}^\top]_1,\ \boxed{[\widetilde{\mathbf{w}}^\top]_1,\ [\widetilde{\mathbf{v}}^\top]_1} \right)$$

and replace $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ with $\mathsf{KeyGen}_3^*(\mathsf{msk}^*, \cdot)$, which works as $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ in $\mathsf{Game}_2$ except that, for the $j$-th query $f_j$, we compute

$$\mathsf{sk}_{f_j}[2] = \begin{pmatrix} \boxed{[\underline{\tilde{\mathbf{t}}}_j^\top - (\mathbf{z}^*)^\top + \widetilde{\mathbf{w}}^\top]_2},\ [\boxed{\tilde{\mathbf{t}}_j^\top} \mathbf{L}_{1,j} + \boxed{\tilde{\mathbf{u}}^\top}(\mathbf{I}_n \otimes \mathbf{R}_j)]_2, \\ [\boxed{\tilde{\mathbf{t}}_j^\top} \mathbf{L}_{0,j} \boxed{-\tilde{\mathbf{u}}^\top (\mathbf{I}_n \otimes \mathbf{R}_j)(\mathbf{x}^* \otimes \mathbf{I}_m) + \widetilde{\mathbf{v}}^\top \mathbf{R}_j}]_2 \end{pmatrix}$$

where $\tilde{\mathbf{w}}, \tilde{\mathbf{v}}$ are given in $\mathsf{msk}^*$ (output by $\mathsf{Setup}^*$) and $\tilde{\mathbf{u}} \leftarrow \mathbb{Z}_p^{kn}$, $\mathbf{t}_j \leftarrow \mathbb{Z}_p^{m+n'-1}$, $\mathbf{R}_j \leftarrow \mathbb{Z}_p^{k\times m}$. We claim that $\mathsf{Game}_2 \approx_s \mathsf{Game}_3$. This follows from the following statement: for any full-rank $(\mathbf{A}|\mathbf{c})$, we have

$$\begin{aligned} &(\mathbf{A}^\top \mathbf{U}, \mathbf{c}^\top \mathbf{U},\ \mathbf{A}^\top \mathbf{W}, \mathbf{c}^\top \mathbf{W},\ \mathbf{A}^\top \mathbf{V}, \mathbf{c}^\top \mathbf{V},\ \mathbf{A}^\top \mathbf{T}_j, \mathbf{c}^\top \mathbf{T}_j) \\ \equiv\ &(\mathbf{A}^\top \mathbf{U}, \boxed{\tilde{\mathbf{u}}^\top},\ \mathbf{A}^\top \mathbf{W}, \boxed{\tilde{\mathbf{w}}^\top - (\mathbf{z}^*)^\top},\ \mathbf{A}^\top \mathbf{V}, \boxed{\tilde{\mathbf{v}}^\top - \tilde{\mathbf{u}}^\top (\mathbf{x}^* \otimes \mathbf{I}_k)},\ \mathbf{A}^\top \mathbf{T}_j, \boxed{\tilde{\mathbf{t}}_j^\top}) \end{aligned}$$

which is implied by Lemma 3.

$\underline{\mathsf{Game}_4}$: Identical to $\mathsf{Game}_3$ except that we replace $\mathsf{KeyGen}_3^*$ with $\mathsf{KeyGen}_4^*$ which works as $\mathsf{KeyGen}_3^*$ except that, for the $j$-th query $f_j$, we compute

$$\mathsf{sk}_{f_j}[2] = \left( [\underline{\tilde{\mathbf{t}}}_j^\top - (\mathbf{z}^*)^\top + \widetilde{\mathbf{w}}^\top]_2,\ [\tilde{\mathbf{t}}_j^\top \mathbf{L}_{1,j} + \boxed{\hat{\mathbf{u}}_j^\top}]_2,\ [\tilde{\mathbf{t}}_j^\top \mathbf{L}_{0,j} - \boxed{\hat{\mathbf{u}}_j^\top}(\mathbf{x}^* \otimes \mathbf{I}_m) + \widetilde{\mathbf{v}}^\top \mathbf{R}_j]_2 \right)$$

where $\hat{\mathbf{u}}_j \leftarrow \mathbb{Z}_p^{nm}$ and $\mathbf{R}_j \leftarrow \mathbb{Z}_p^{k\times m}$. We claim that $\mathsf{Game}_3 \approx_c \mathsf{Game}_4$. This follows from $\mathrm{MDDH}_{k,mQ}^n$ assumption which tells us that

$$\left\{ [\widetilde{\mathbf{u}}^\top (\mathbf{I}_n \otimes \mathbf{R}_j)]_2,\ [\mathbf{R}_j]_2 \right\}_{j\in[Q]} \approx_c \left\{ \boxed{[\hat{\mathbf{u}}_j^\top]_2},\ [\mathbf{R}_j]_2 \right\}_{j\in[Q]}$$

where $Q$ is the number of key queries.

$\underline{\mathsf{Game}_5}$: Identical to $\mathsf{Game}_4$ except that we replace $\mathsf{KeyGen}_4^*$ with $\mathsf{KeyGen}^*$; this is the ideal game. We claim that $\mathsf{Game}_4 \approx_s \mathsf{Game}_5$. This follows from the privacy of partial garbling scheme in Sect. 4.2.

We prove the indistinguishability of adjacent games listed above in the full paper.

## 6 $\Pi_{\mathsf{ext}}$: Extending $\Pi_{\mathsf{one}}$

In this section, we extend our one-slot FE scheme $\Pi_{\mathsf{one}}$ in Sect. 5 to handle the randomization offsets $\mathbf{w}^\top\mathbf{r}$. The scheme achieves simulation-based semi-adaptive security under $k$-Linear assumption.

**Extension.** The extended scheme is the same as a one-slot FE for attribute-weighted sums, except we replace functionality $((\mathbf{x}, \mathbf{z}), f) \mapsto f(\mathbf{x})^\top\mathbf{z}$ with

$$((\mathbf{x}, \mathbf{z}\|\mathbf{w}), (f, [\mathbf{r}]_2)) \mapsto [f(\mathbf{x})^\top\mathbf{z} + \mathbf{w}^\top\mathbf{r}]_T$$

where $\mathbf{w}, \mathbf{r} \in \mathbb{Z}_p^k$. That is, we make the following modifications:

- $\mathsf{Enc}$ takes $\mathbf{z}\|\mathbf{w}$ instead of $\mathbf{z}$ as the second input;
- $\mathsf{KeyGen}, \mathsf{KeyGen}^*$ takes $(f, [\mathbf{r}]_2)$ instead of $f$ as input;
- in correctness, decryption computes $[f(\mathbf{x})^\top\mathbf{z} + \mathbf{w}^\top\mathbf{r}]_T$ instead of $f(\mathbf{x})^\top\mathbf{z}$;
- in the security definition, $\mathcal{A}$ produces $(\mathbf{x}^*, \mathbf{z}^*\|\mathbf{w}^*)$ instead of $(\mathbf{x}^*, \mathbf{z}^*)$, and $\mathsf{KeyGen}^*$ gets $[f(\mathbf{x}^*)^\top\mathbf{z}^* + (\mathbf{w}^*)^\top\mathbf{r}]_2$ instead of $f(\mathbf{x}^*)^\top\mathbf{z}^*$.

In particular, correctness states that:

$$\mathsf{Dec}(\mathsf{Enc}(\mathsf{mpk}, (\mathbf{x}, \mathbf{z}\|\mathbf{w})), \mathsf{KeyGen}(\mathsf{msk}, (f, [\mathbf{r}]_2))) = [f(\mathbf{x})^\top\mathbf{z} + \mathbf{w}^\top\mathbf{r}]_T$$

**Construction overview.** To obtain a scheme with the extension, the idea — following the IPFE in [6]— is to augment the previous construction $\Pi_{\mathsf{one}}$ with $[\mathbf{A}^\top\mathbf{W}_0]_1$ in $\mathsf{mpk}$, $[\mathbf{w}^\top + \mathbf{s}^\top\mathbf{A}^\top\mathbf{W}_0]_1$ in the ciphertext, and $[\mathbf{W}_0\mathbf{r}]_2$ in the secret key. During decryption, we will additionally compute

$$e([\mathbf{w}^\top + \mathbf{s}^\top\mathbf{A}^\top\mathbf{W}_0]_1, [\mathbf{r}]_2) \cdot e([\mathbf{s}^\top\mathbf{A}^\top]_1, [\mathbf{W}_0\mathbf{r}]_2)^{-1} = [\mathbf{w}^\top\mathbf{r}]_T$$

This works for correctness, but violates security since the decryptor learns both $[f(\mathbf{x})^\top\mathbf{z}]_T$ and $[\mathbf{w}^\top\mathbf{r}]_T$ instead of just the sum. To avoid this leakage while preserving correctness, we will carefully embed $\mathbf{W}_0\mathbf{r}$ into the secret key for $\Pi_{\mathsf{one}}$, while relying on the extension of the garbling scheme for handling shifts to argue both correctness and security, cf. Sect. 4.2. We will describe the scheme and simulator but defer the details for the proof to full paper.

### 6.1 Our Scheme

**Scheme.** Our extended one-slot FE scheme $\Pi_{\mathsf{ext}}$ in prime-order bilinear group is described as follows. The boxes indicate the changes from the scheme in Sect. 5.1.

– $\mathsf{Setup}(1^\lambda, 1^n, 1^{n'})$: Run $\mathbb{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$. Sample $\mathbf{A} \leftarrow \mathbb{Z}_p^{(k+1)\times k}$ and

$$\mathbf{W} \leftarrow \mathbb{Z}_p^{(k+1)\times n'}, \boxed{\mathbf{W}_0 \leftarrow \mathbb{Z}_p^{(k+1)\times k}}, \mathbf{U} \leftarrow \mathbb{Z}_p^{(k+1)\times kn}, \mathbf{V} \leftarrow \mathbb{Z}_p^{(k+1)\times k}$$

and output

$$\mathsf{mpk} = \big(\mathbb{G}, [\mathbf{A}^\top]_1, [\mathbf{A}^\top\mathbf{W}]_1, [\mathbf{A}^\top\mathbf{U}]_1, [\mathbf{A}^\top\mathbf{V}]_1, \boxed{[\mathbf{A}^\top\mathbf{W}_0]_1}\big)$$
$$\mathsf{msk} = \big(\mathbf{W}, \mathbf{U}, \mathbf{V}, \boxed{\mathbf{W}_0}\big).$$

– $\mathsf{Enc}(\mathsf{mpk}, (\mathbf{x}, \mathbf{z}\|\mathbf{w}))$: Sample $\mathbf{s} \leftarrow \mathbb{Z}_p^k$ and output

$$\mathsf{ct}_{\mathbf{x},\mathbf{z}\|\mathbf{w}} = \left( \begin{array}{c} [\mathbf{s}^\top\mathbf{A}^\top]_1, [\mathbf{z}^\top + \mathbf{s}^\top\mathbf{A}^\top\mathbf{W}]_1, [\mathbf{s}^\top\mathbf{A}^\top\mathbf{U}(\mathbf{x}\otimes\mathbf{I}_k) + \mathbf{s}^\top\mathbf{A}^\top\mathbf{V}]_1, \\ \boxed{[\mathbf{w}^\top + \mathbf{s}^\top\mathbf{A}^\top\mathbf{W}_0]_1} \end{array} \right), \mathbf{x}.$$

– $\mathsf{KeyGen}(\mathsf{msk}, (f, [\mathbf{r}]_2))$: Run $(\mathbf{L}_1, \mathbf{L}_0) \leftarrow \mathsf{lgen}(f)$ where $\mathbf{L}_1 \in \mathbb{Z}_p^{(m+n'-1)\times mn}$, $\mathbf{L}_0 \in \mathbb{Z}_p^{(m+n'-1)\times m}$ (cf. Sect. 4.2). Sample $\mathbf{T} \leftarrow \mathbb{Z}_p^{(k+1)\times(m+n'-1)}$ and $\mathbf{R} \leftarrow \mathbb{Z}_p^{k\times m}$ and output[6]

$$\mathsf{sk}_{f,\mathbf{r}} = \big([\underline{\mathbf{T}} + \mathbf{W}]_2, [\mathbf{T}\mathbf{L}_1 + \mathbf{U}(\mathbf{I}_n \otimes \mathbf{R})]_2, [\mathbf{T}\mathbf{L}_0 - \boxed{\mathbf{W}_0\mathbf{r}\cdot\mathbf{e}_1^\top} + \mathbf{V}\mathbf{R}]_2, [\mathbf{R}]_2\big), (f, \boxed{[\mathbf{r}]_2})$$

where $\underline{\mathbf{T}}$ refers to the matrix composed of the right most $n'$ columns of $\mathbf{T}$.

– $\mathsf{Dec}((\mathsf{sk}_{f,\mathbf{r}}, (f, \boxed{[\mathbf{r}]_2})), (\mathsf{ct}_{\mathbf{x},\mathbf{z}\|\mathbf{w}}, \mathbf{x}))$: On input key:

$$\mathsf{sk}_{f,\mathbf{r}} = \big([\mathbf{K}_1]_2, [\mathbf{K}_2]_2, [\mathbf{K}_3]_2, [\mathbf{R}]_2\big) \quad \text{and} \quad (f, [\mathbf{r}]_2)$$

and ciphertext:

$$\mathsf{ct}_{\mathbf{x},\mathbf{z}\|\mathbf{w}} = \big([\mathbf{c}_0^\top]_1, [\mathbf{c}_1^\top]_1, [\mathbf{c}_2^\top]_1, [\mathbf{c}_3^\top]_1\big) \quad \text{and} \quad \mathbf{x}$$

the decryption works as follows:

1. compute
$$[\mathbf{p}_1^\top]_T = e([\mathbf{c}_1^\top]_1, [\mathbf{I}_{n'}]_2) \cdot e([\mathbf{c}_0^\top]_1, [-\mathbf{K}_1]_2) \tag{16}$$
2. compute
$$[\mathbf{p}_2^\top]_T = e([\mathbf{c}_0^\top]_1, [\mathbf{K}_2(\mathbf{x}\otimes\mathbf{I}_m) + \mathbf{K}_3]_2) \cdot e([-\mathbf{c}_2^\top]_1, [\mathbf{R}]_2) \cdot \boxed{e([\mathbf{c}_3^\top]_1, [\mathbf{r}\cdot\mathbf{e}_1^\top]_2)} \tag{17}$$
3. run $\mathbf{d}_{f,\mathbf{x}} \leftarrow \mathsf{rec}(f, \mathbf{x})$ (see Sect. 4.2), output
$$[D]_T = [(\mathbf{p}_1^\top, \mathbf{p}_2^\top)\mathbf{d}_{f,\mathbf{x}}]_T \tag{18}$$

[6] We use $\mathbf{r}$ instead of $[\mathbf{r}]_2$ in the subscript here and note that the function is described by $(f, [\mathbf{r}]_2)$ rather than $(f, \mathbf{r})$.

**Simulator.** The simulator for $\Pi_{\mathsf{ext}}$ is as follows. The boxes indicate the changes from the simulator for $\Pi_{\mathsf{one}}$ in Sect. 5.2.

– $\mathsf{Setup}^*(1^\lambda, 1^n, 1^{n'})$: Run $\mathbb{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$. Sample

$$\mathbf{A} \leftarrow \mathbb{Z}_p^{(k+1)\times k} \quad \text{and} \quad \mathbf{c} \leftarrow \mathbb{Z}_p^{k+1} \quad \text{and}$$

$$\mathbf{W} \leftarrow \mathbb{Z}_p^{(k+1)\times n'}, \boxed{\mathbf{W}_0 \leftarrow \mathbb{Z}_p^{(k+1)\times k}}, \mathbf{U} \leftarrow \mathbb{Z}_p^{(k+1)\times kn}, \mathbf{V} \leftarrow \mathbb{Z}_p^{(k+1)\times k}$$
$$\widetilde{\mathbf{w}} \leftarrow \mathbb{Z}_p^{n'}, \quad \boxed{\widetilde{\mathbf{w}}_0 \leftarrow \mathbb{Z}_p^{k}}, \quad \widetilde{\mathbf{v}} \leftarrow \mathbb{Z}_p^{k}$$

and output

$$\mathsf{mpk} = \big(\mathbb{G}, [\mathbf{A}^\top]_1, [\mathbf{A}^\top\mathbf{W}]_1, \boxed{[\mathbf{A}^\top\mathbf{W}_0]_1}, [\mathbf{A}^\top\mathbf{U}]_1, [\mathbf{A}^\top\mathbf{V}]_1\big)$$
$$\mathsf{msk}^* = \big(\mathbf{W}, \boxed{\mathbf{W}_0}, \mathbf{U}, \mathbf{V}, \widetilde{\mathbf{w}}, \boxed{\widetilde{\mathbf{w}}_0}, \widetilde{\mathbf{v}}, \mathbf{c}, \mathbf{C}^\perp, \mathbf{A}, \mathbf{a}^\perp\big)$$

where $(\mathbf{A}|\mathbf{c})^\top(\mathbf{C}^\perp|\mathbf{a}^\perp) = \mathbf{I}_{k+1}$. Here we assume that $(\mathbf{A}|\mathbf{c})$ has full rank, which happens with probability $1 - 1/p$.

– $\mathsf{Enc}^*(\mathsf{msk}^*, \mathbf{x}^*)$: Output

$$\mathsf{ct}^* = \big([\mathbf{c}^\top]_1, [\widetilde{\mathbf{w}}^\top]_1, [\widetilde{\mathbf{v}}^\top]_1, \boxed{[\widetilde{\mathbf{w}}_0^\top]_1}\big) \quad \text{and} \quad \mathbf{x}^*.$$

– $\mathsf{KeyGen}^*(\mathsf{msk}^*, \mathbf{x}^*, (f, [\mathbf{r}]_2), [\mu]_2)$: Run

$$(\mathbf{L}_1, \mathbf{L}_0) \leftarrow \mathsf{lgen}(f) \quad \text{and} \quad ([(\mathbf{p}_1^*)^\top]_2, [(\mathbf{p}_2^*)^\top]_2) \leftarrow \mathsf{pgb}^*(f, \mathbf{x}^*, \boxed{[\mu]_2}).$$

Here, we use the fact that $\mathsf{pgb}^*(f, \mathbf{x}^*, \cdot)$ is an affine function. Sample $\hat{\mathbf{u}} \leftarrow \mathbb{Z}_p^{nm}$, $\mathbf{T} \leftarrow \mathbb{Z}_p^{(k+1)\times(m+n'-1)}$ and $\mathbf{R} \leftarrow \mathbb{Z}_p^{k\times m}$ and output

$$\mathsf{sk}^*_{f,\mathbf{r}} = \big(\mathbf{C}^\perp \cdot \mathsf{sk}^*_{f,\mathbf{r}}[1] + \mathbf{a}^\perp \cdot \mathsf{sk}^*_{f,\mathbf{r}}[2], [\mathbf{R}]_2\big) \quad \text{and} \quad (f, [\mathbf{r}]_2) \tag{19}$$

where

$$\mathsf{sk}^*_{f,\mathbf{r}}[1] = \big([\mathbf{A}^\top\underline{\mathbf{T}} + \mathbf{A}^\top\mathbf{W}]_2, [\mathbf{A}^\top\mathbf{T}\mathbf{L}_1 + \mathbf{A}^\top\mathbf{U}(\mathbf{I}_n \otimes \mathbf{R})]_2,$$
$$[\mathbf{A}^\top\mathbf{T}\mathbf{L}_0 - \boxed{\mathbf{A}^\top\mathbf{W}_0\mathbf{r}\cdot\mathbf{e}_1^\top} + \mathbf{A}^\top\mathbf{V}\mathbf{R}]_2\big)$$
$$\mathsf{sk}^*_{f,\mathbf{r}}[2] = \big([-(\mathbf{p}_1^*)^\top + \widetilde{\mathbf{w}}^\top]_2, [\hat{\mathbf{u}}^\top]_2, [(\mathbf{p}_2^*)^\top - \hat{\mathbf{u}}^\top(\mathbf{x}^* \otimes \mathbf{I}_m) - \boxed{\widetilde{\mathbf{w}}_0^\top\mathbf{r}\cdot\mathbf{e}_1^\top} + \widetilde{\mathbf{v}}^\top\mathbf{R}]_2\big)$$

Here $\underline{\mathbf{T}}$ refers to the matrix composed of the right most $n'$ columns of $\mathbf{T}$. That is,

$$\mathsf{sk}^*_{f,\mathbf{r}} = \begin{pmatrix} [\mathbf{C}^\perp(\mathbf{A}^\top\underline{\mathbf{T}} + \mathbf{A}^\top\mathbf{W}) & +\mathbf{a}^\perp(-(\mathbf{p}_1^*)^\top + \widetilde{\mathbf{w}}^\top)]_2, \\ [\mathbf{C}^\perp(\mathbf{A}^\top\mathbf{T}\mathbf{L}_1 + \mathbf{A}^\top\mathbf{U}(\mathbf{I}_n \otimes \mathbf{R})) & +\mathbf{a}^\perp(\hat{\mathbf{u}}^\top)]_2 \\ [\mathbf{C}^\perp(\mathbf{A}^\top\mathbf{T}\mathbf{L}_0 - \boxed{\mathbf{A}^\top\mathbf{W}_0\mathbf{r}\cdot\mathbf{e}_1^\top} + \mathbf{A}^\top\mathbf{V}\mathbf{R}) & +\mathbf{a}^\perp((\mathbf{p}_2^*)^\top - \hat{\mathbf{u}}^\top(\mathbf{x}^* \otimes \mathbf{I}_m) - \boxed{\widetilde{\mathbf{w}}_0^\top\mathbf{r}\cdot\mathbf{e}_1^\top} + \widetilde{\mathbf{v}}^\top\mathbf{R})]_2 \end{pmatrix}, [\mathbf{R}]_2$$

## 7 $\Pi_{\mathsf{ubd}}$: Unbounded-Slot Scheme

In this section, we describe our unbounded-slot FE scheme. We give a generic transformation from scheme $\Pi_{\mathsf{ext}}$ in Sect. 6 and present a self-contained description of the scheme in the full paper.

### 7.1 Scheme

Let $\Pi_{\mathsf{ext}} = (\mathsf{Setup}_{\mathsf{ext}}, \mathsf{Enc}_{\mathsf{ext}}, \mathsf{KeyGen}_{\mathsf{ext}}, \mathsf{Dec}_{\mathsf{ext}})$ be the extended one-slot FE scheme in Sect. 6. Our unbounded-slot FE scheme $\Pi_{\mathsf{ubd}}$ is as follows:

– $\mathsf{Setup}(1^\lambda, 1^n, 1^{n'})$: Run

$$(\mathsf{mpk}_1, \mathsf{msk}_1) \leftarrow \mathsf{Setup}_{\mathsf{ext}}(1^\lambda, 1^n, 1^{n'}); \quad (\mathsf{mpk}_2, \mathsf{msk}_2) \leftarrow \mathsf{Setup}_{\mathsf{ext}}(1^\lambda, 1^n, 1^{n'})$$

and output

$$\mathsf{mpk} = (\mathsf{mpk}_1, \mathsf{mpk}_2) \quad \text{and} \quad \mathsf{msk} = (\mathsf{msk}_1, \mathsf{msk}_2).$$

– $\mathsf{Enc}(\mathsf{mpk}, (\mathbf{x}_i, \mathbf{z}_i)_{i\in[N]})$: Sample $\mathbf{w}_2, \ldots, \mathbf{w}_N \leftarrow \mathbb{Z}_p^k$, compute

$$\begin{aligned} \mathsf{ct}_1 &\leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_1, (\mathbf{x}_1, \mathbf{z}_1 \| -\textstyle\sum_{i\in[2,N]} \mathbf{w}_i)) \\ \mathsf{ct}_i &\leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}_i, \mathbf{z}_i \| \mathbf{w}_i)), \quad \forall i \in [2, N] \end{aligned}$$

and output

$$\mathsf{ct}_{(\mathbf{x}_i, \mathbf{z}_i)} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_N) \quad \text{and} \quad (\mathbf{x}_i)_{i\in[N]}.$$

– $\mathsf{KeyGen}(\mathsf{msk}, f)$: Pick $\mathbf{r} \leftarrow \mathbb{Z}_p^k$, compute

$$\mathsf{sk}_{f,1} \leftarrow \mathsf{KeyGen}_{\mathsf{ext}}(\mathsf{msk}_1, (f, [\mathbf{r}]_2)); \qquad \mathsf{sk}_{f,2} \leftarrow \mathsf{KeyGen}_{\mathsf{ext}}(\mathsf{msk}_2, (f, [\mathbf{r}]_2))$$

and output

$$\mathsf{sk}_f = (\mathsf{sk}_{f,1}, \mathsf{sk}_{f,2}, [\mathbf{r}]_2) \quad \text{and} \quad f.$$

– $\mathsf{Dec}((\mathsf{sk}_f, f), (\mathsf{ct}_{(\mathbf{x}_i, \mathbf{z}_i)}, (\mathbf{x}_i)_{i\in[N]}))$: Parse ciphertext and key as

$$\mathsf{sk}_f = (\mathsf{sk}_{f,1}, \mathsf{sk}_{f,2}, [\mathbf{r}]_2) \quad \text{and} \quad \mathsf{ct}_{(\mathbf{x}_i, \mathbf{z}_i)} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_N).$$

We proceed as follows:

1. Compute
$$[D_1]_T \leftarrow \mathsf{Dec}_{\mathsf{ext}}\big((\mathsf{sk}_{f,1}, (f, [\mathbf{r}]_2)), (\mathsf{ct}_1, \mathbf{x}_1)\big); \tag{20}$$
2. For all $i \in [2, N]$, compute
$$[D_i]_T \leftarrow \mathsf{Dec}_{\mathsf{ext}}\big((\mathsf{sk}_{f,2}, (f, [\mathbf{r}]_2)), (\mathsf{ct}_i, \mathbf{x}_i)\big); \tag{21}$$
3. Compute
$$[D]_T = [D_1]_T \cdots [D_N]_T \tag{22}$$
and output $D$ via brute-force discrete log.

**Correctness.** For $\mathsf{ct}_{(\mathbf{x}_i,\mathbf{z}_i)}$ with randomness $\mathbf{w}_2,\dots,\mathbf{w}_N$ and $\mathsf{sk}_f$ with randomness $\mathbf{r}$, we have

$$D_1 = f(\mathbf{x}_1)^\top \mathbf{z}_1 - \textstyle\sum_{i\in[2,N]} \mathbf{w}_i^\top \mathbf{r} \tag{23}$$

$$D_i = f(\mathbf{x}_i)^\top \mathbf{z}_i + \mathbf{w}_i^\top \mathbf{r}, \qquad \forall i \in [2,N] \tag{24}$$

$$D = \textstyle\sum_{i\in[N]} f(\mathbf{x}_i)^\top \mathbf{z}_i \tag{25}$$

Here (23) and (24) follow from the correctness of $\Pi_{\mathsf{ext}}$ and the last (25) is implied by (23) and (24). This readily proves the correctness.

**Security.** We have the following theorem with the proof shown in the subsequent subsection.

**Theorem 3.** *Assume that extended one-slot scheme $\Pi_{\mathsf{ext}}$ achieves simulation-based semi-adaptive security, our unbounded-slot FE scheme $\Pi_{\mathsf{ubd}}$ described in this section achieves simulation-based semi-adaptive security under the $k$-Linear assumption in $\mathbb{G}_2$.*

### 7.2 Simulator

Let $(\mathsf{Setup}^*_{\mathsf{ext}}, \mathsf{Enc}^*_{\mathsf{ext}}, \mathsf{KeyGen}^*_{\mathsf{ext}})$ be the simulator for $\Pi_{\mathsf{ext}}$, we start by describing the simulator for $\Pi_{\mathsf{ubd}}$. As written, the adversary needs to commit to the length $N$ in advance; this is merely an artifact of our formalization of simulation-based security, and can be avoided by having $\mathsf{Enc}^*$ pass auxiliary information to $\mathsf{KeyGen}^*$.

- $\mathsf{Setup}^*(1^\lambda, 1^n, 1^{n'}, 1^N)$: Sample $\mathbf{w}_2,\dots,\mathbf{w}_N \leftarrow \mathbb{Z}_p^k$, run

  $$(\mathsf{mpk}_1, \mathsf{msk}_1^*) \leftarrow \mathsf{Setup}^*_{\mathsf{ext}}(1^\lambda, 1^n, 1^{n'}); \ (\mathsf{mpk}_2, \mathsf{msk}_2) \leftarrow \mathsf{Setup}_{\mathsf{ext}}(1^\lambda, 1^n, 1^{n'})$$

  and output

  $$\mathsf{mpk} = (\mathsf{mpk}_1, \mathsf{mpk}_2) \quad \text{and} \quad \mathsf{msk}^* = (\mathsf{msk}_1^*, \mathsf{msk}_2, \mathbf{w}_2, \dots, \mathbf{w}_N).$$

- $\mathsf{Enc}^*(\mathsf{msk}^*, (\mathbf{x}_i^*)_{i\in[N]})$: Compute

  $$\mathsf{ct}_1^* \leftarrow \mathsf{Enc}^*_{\mathsf{ext}}(\mathsf{msk}_1^*, \mathbf{x}_1^*) \quad \text{and} \quad \mathsf{ct}_i \leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}_i^*, \mathbf{0}\|\mathbf{w}_i)), \ \forall i \in [2,N]$$

  and output

  $$\mathsf{ct}^* = (\mathsf{ct}_1^*, \mathsf{ct}_2, \dots, \mathsf{ct}_N) \quad \text{and} \quad (\mathbf{x}_i^*)_{i\in[N]}.$$

- $\mathsf{KeyGen}^*(\mathsf{msk}^*, (\mathbf{x}_i^*)_{i\in[N]}, f, \mu \in \mathbb{Z}_p)$: Pick $\mathbf{r} \leftarrow \mathbb{Z}_p^k$, compute

  $$\mathsf{sk}^*_{f,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{ext}}(\mathsf{msk}_1^*, \mathbf{x}_1^*, (f, [\mathbf{r}]_2), [\mu - \textstyle\sum_{i\in[2,N]} \mathbf{w}_i^\top \mathbf{r}]_2)$$
  $$\mathsf{sk}_{f,2} \leftarrow \mathsf{KeyGen}_{\mathsf{ext}}(\mathsf{msk}_2, (f, [\mathbf{r}]_2))$$

  and output

  $$\mathsf{sk}_f^* = (\mathsf{sk}^*_{f,1}, \mathsf{sk}_{f,2}, [\mathbf{r}]_2) \quad \text{and} \quad f.$$

### 7.3 Proof

With our simulator, we prove the following theorem which implies Theorem 3.

**Theorem 4.** *For all $\mathcal{A}$, there exist $\mathcal{B}_1$ and $\mathcal{B}_2$ with $\mathsf{Time}(\mathcal{B}_1), \mathsf{Time}(\mathcal{B}_2) \approx \mathsf{Time}(\mathcal{A})$ such that*

$$\mathsf{Adv}^{\Pi_{\mathsf{ubd}}}_{\mathcal{A}}(\lambda) \leq (2N-1) \cdot \mathsf{Adv}^{\Pi_{\mathsf{ext}}}_{\mathcal{B}_1}(\lambda) + (N-1) \cdot \mathsf{Adv}^{\mathrm{MDDH}^1_{k,Q}}_{\mathcal{B}_2}(\lambda)$$

*where $Q$ is the number of key queries and $N$ is number of slots.*

**Game sequence.** We use $(\mathbf{x}^*_1, \mathbf{z}^*_1, \ldots, \mathbf{x}^*_N, \mathbf{z}^*_N)$ to denote the semi-adaptive challenge and prove Theorem 4 via the following game sequence summarized in Fig. 4, where

$$\begin{aligned}\mathsf{Game}_0 \approx_c \mathsf{Game}_1 &= \mathsf{Game}_{2.0} \approx_c \mathsf{Game}_{2.1} \approx_c \mathsf{Game}_{2.2} \approx_c \mathsf{Game}_{2.3} \\ &\cdots \\ &= \mathsf{Game}_{N.0} \approx_c \mathsf{Game}_{N.1} \approx_c \mathsf{Game}_{N.2} \approx_c \mathsf{Game}_{N.3}\end{aligned}$$

$\underline{\mathsf{Game}_0}$: Real game.
$\underline{\mathsf{Game}_1}$: Identical to $\mathsf{Game}_0$ except for the boxed terms below:

- we generate $\mathsf{mpk} = (\mathsf{mpk}_1, \mathsf{mpk}_2)$ and $\mathsf{msk} = (\boxed{\mathsf{msk}^*_1}, \mathsf{msk}_2)$ where

$$\boxed{(\mathsf{mpk}_1, \mathsf{msk}^*_1) \leftarrow \mathsf{Setup}^*_{\mathsf{ext}}(1^\lambda, 1^n, 1^{n'})}; \quad (\mathsf{mpk}_2, \mathsf{msk}_2) \leftarrow \mathsf{Setup}_{\mathsf{ext}}(1^\lambda, 1^n, 1^{n'})$$

- the challenge ciphertext for $(\mathbf{x}^*_1, \mathbf{z}^*_1, \ldots, \mathbf{x}^*_N, \mathbf{z}^*_N)$ is $\mathsf{ct}^* = (\boxed{\mathsf{ct}^*_1}, \mathsf{ct}_2, \ldots, \mathsf{ct}_N)$ where

$$\boxed{\mathsf{ct}^*_1 \leftarrow \mathsf{Enc}^*_{\mathsf{ext}}(\mathsf{msk}^*_1, \mathbf{x}^*_1)}; \quad \mathsf{ct}_i \leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}^*_i, \mathbf{z}^*_i \| \mathbf{w}_i)),\ \forall i \in [2, N]$$

- the key for the $j$-th query $f_j$ is $\mathsf{sk}_{f_j} = (\boxed{\mathsf{sk}^*_{f_j,1}}, \mathsf{sk}_{f_j,2}, [\mathbf{r}_j]_2)$ where

$$\boxed{\mathsf{sk}^*_{f_j,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{ext}}\big(\mathsf{msk}^*_1, \mathbf{x}^*_1, (f_j, [\mathbf{r}_j]_2), [f_j(\mathbf{x}^*_1)^\top \mathbf{z}^*_1 - \textstyle\sum_{i\in[2,N]} \mathbf{w}_i^\top \mathbf{r}_j]_2\big)}$$

$$\mathsf{sk}_{f_j,2} \leftarrow \mathsf{KeyGen}_{\mathsf{ext}}(\mathsf{msk}_2, (f_j, [\mathbf{r}_j]_2));$$

where $\mathbf{w}_2, \ldots, \mathbf{w}_N \leftarrow \mathbb{Z}_p^k$ and $\mathbf{r}_j \leftarrow \mathbb{Z}_p^k$ for all $j \in [Q]$. We claim that $\mathsf{Game}_0 \approx_c \mathsf{Game}_1$. This follows from the simulation-based semi-adaptive security of $\Pi_{\mathsf{ext}}$.

$\underline{\mathsf{Game}_{\eta.0}}$ for $\eta \in [2, N]$: Identical to $\mathsf{Game}_1$ except for the boxed terms below:

- the challenge ciphertext for $(\mathbf{x}^*_1, \mathbf{z}^*_1, \ldots, \mathbf{x}^*_N, \mathbf{z}^*_N)$ is $\mathsf{ct}^* = (\mathsf{ct}^*_1, \mathsf{ct}_2, \ldots, \mathsf{ct}_N)$ where

$$\mathsf{ct}^*_1 \leftarrow \mathsf{Enc}^*_{\mathsf{ext}}(\mathsf{msk}^*_1, \mathbf{x}^*_1); \quad \mathsf{ct}_i \leftarrow \begin{cases} \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}^*_i, \boxed{\mathbf{0}} \| \mathbf{w}_i)) & i \in [2, \eta-1] \\ \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}^*_i, \mathbf{z}^*_i \| \mathbf{w}_i)) & i \in [\eta, N] \end{cases}$$

– the key for the $j$-th query $f_j$ is $\mathsf{sk}_{f_j} = (\mathsf{sk}^*_{f_j,1}, \mathsf{sk}_{f_j,2}, [\mathbf{r}_j]_2)$ where

$$\mathsf{sk}^*_{f_j,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{ext}}\big(\mathsf{msk}^*_1, \mathbf{x}^*_1, (f_j, [\mathbf{r}_j]_2), [\boxed{\textstyle\sum_{i\in[\eta-1]} f_j(\mathbf{x}^*_i)^\top \mathbf{z}^*_i} - \textstyle\sum_{i\in[2,N]} \mathbf{w}_i^\top \mathbf{r}_j]_2\big)$$

$$\mathsf{sk}_{f_j,2} \leftarrow \mathsf{KeyGen}_{\mathsf{ext}}(\mathsf{msk}_2, (f_j, [\mathbf{r}_j]_2));$$

where $\mathbf{w}_2, \ldots, \mathbf{w}_N \leftarrow \mathbb{Z}_p^k$ and $\mathbf{r}_j \leftarrow \mathbb{Z}_p^k$ for all $j \in [Q]$.

$\underline{\mathsf{Game}_{\eta.1}}$ for $\eta \in [2, N]$: Identical to $\mathsf{Game}_{\eta.0}$ except for the boxed terms below:

– we generate $\mathsf{mpk} = (\mathsf{mpk}_1, \mathsf{mpk}_2)$ and $\mathsf{msk} = (\mathsf{msk}^*_1, \boxed{\mathsf{msk}^*_2})$ where

$$(\mathsf{mpk}_1, \mathsf{msk}^*_1) \leftarrow \mathsf{Setup}^*_{\mathsf{ext}}(1^\lambda, 1^n, 1^{n'}); \quad \boxed{(\mathsf{mpk}_2, \mathsf{msk}^*_2) \leftarrow \mathsf{Setup}^*_{\mathsf{ext}}(1^\lambda, 1^n, 1^{n'})}$$

– the challenge ciphertext for $(\mathbf{x}^*_1, \mathbf{z}^*_1, \ldots, \mathbf{x}^*_N, \mathbf{z}^*_N)$ is $\mathsf{ct}^* = (\mathsf{ct}^*_1, \mathsf{ct}_2, \ldots, \mathsf{ct}_{\eta-1}, \boxed{\mathsf{ct}^*_\eta}, \mathsf{ct}_{\eta+1}, \ldots, \mathsf{ct}_N)$ where

$$\mathsf{ct}^*_1 \leftarrow \mathsf{Enc}^*_{\mathsf{ext}}(\mathsf{msk}^*_1, \mathbf{x}^*_1), \begin{cases} \mathsf{ct}_i \leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}^*_i, \mathbf{0} \| \mathbf{w}_i)) & i \in [2, \eta-1] \\ \boxed{\mathsf{ct}^*_\eta \leftarrow \mathsf{Enc}^*_{\mathsf{ext}}(\mathsf{msk}^*_2, \mathbf{x}^*_\eta)} & i = \eta \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}^*_i, \mathbf{z}^*_i \| \mathbf{w}_i)) & i \in [\eta+1, N] \end{cases}$$

– the key for the $j$-th query $f_j$ is $\mathsf{sk}_{f_j} = (\mathsf{sk}^*_{f_j,1}, \boxed{\mathsf{sk}^*_{f_j,2}}, [\mathbf{r}_j]_2)$ where

$$\mathsf{sk}^*_{f_j,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{ext}}\big(\mathsf{msk}^*_1, \mathbf{x}^*_1, (f_j, [\mathbf{r}_j]_2), [\textstyle\sum_{i\in[\eta-1]} f_j(\mathbf{x}^*_i)^\top \mathbf{z}^*_i - \sum_{i\in[2,N]} \mathbf{w}_i^\top \mathbf{r}_j]_2\big)$$

$$\boxed{\mathsf{sk}^*_{f_j,2} \leftarrow \mathsf{KeyGen}^*_{\mathsf{ext}}(\mathsf{msk}^*_2, \mathbf{x}^*_\eta, (f_j, [\mathbf{r}_j]_2), [f_j(\mathbf{x}^*_\eta)^\top \mathbf{z}^*_\eta + \mathbf{w}_\eta^\top \mathbf{r}_j]_2)}$$

where $\mathbf{w}_2, \ldots, \mathbf{w}_N \leftarrow \mathbb{Z}_p^k$ and $\mathbf{r}_j \leftarrow \mathbb{Z}_p^k$ for all $j \in [Q]$. We claim that $\mathsf{Game}_{\eta.0} \approx_c \mathsf{Game}_{\eta.1}$. This follows from the simulation-based semi-adaptive security of $\Pi_{\mathsf{ext}}$.

$\underline{\mathsf{Game}_{\eta.2}}$ for $\eta \in [2, N]$: Identical to $\mathsf{Game}_{\eta.1}$ except for the boxed terms below:

– the key for the $j$-th query $f_j$ is $\mathsf{sk}_{f_j} = (\mathsf{sk}^*_{f_j,1}, \mathsf{sk}^*_{f_j,2}, [\mathbf{r}_j]_2)$ where

$$\mathsf{sk}^*_{f_j,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{ext}}\big(\mathsf{msk}^*_1, \mathbf{x}^*_1, (f_j, [\mathbf{r}_j]_2), [\boxed{\textstyle\sum_{i\in[\eta]} f_j(\mathbf{x}^*_i)^\top \mathbf{z}^*_i} - \textstyle\sum_{i\in[2,N]} \mathbf{w}_i^\top \mathbf{r}_j]_2\big)$$

$$\mathsf{sk}^*_{f_j,2} \leftarrow \mathsf{KeyGen}^*_{\mathsf{ext}}(\mathsf{msk}^*_2, \mathbf{x}^*_\eta, (f_j, [\mathbf{r}_j]_2), [\boxed{\mathbf{w}_\eta^\top \mathbf{r}_j}]_2)$$

where $\mathbf{w}_2, \ldots, \mathbf{w}_N \leftarrow \mathbb{Z}_p^k$ and $\mathbf{r}_j \leftarrow \mathbb{Z}_p^k$ for all $j \in [Q]$. We claim that $\mathsf{Game}_{\eta.1} \approx_c \mathsf{Game}_{\eta.2}$. This follows from Lemma 1 w.r.t. $\mathbf{w}_\eta$ and $f_j(\mathbf{x}^*_\eta)^\top \mathbf{z}^*_\eta$ which is implied by $\mathrm{MDDH}^1_{k,Q}$ assumption: for all $f_j, \mathbf{x}^*_\eta, \mathbf{z}^*_\eta$,

$$\begin{aligned} &\Big\{ \overbrace{[-\mathbf{w}_\eta^\top \mathbf{r}_j]_2}^{\mathsf{sk}^*_{f_j,1}}, \overbrace{[\boxed{f_j(\mathbf{x}^*_\eta)^\top \mathbf{z}^*_\eta} + \mathbf{w}_\eta^\top \mathbf{r}_j]_2}^{\mathsf{sk}^*_{f_j,2}}, [\mathbf{r}_j]_2 \Big\}_{j\in[Q]} \\ \approx_c &\Big\{ [\boxed{f_j(\mathbf{x}^*_\eta)^\top \mathbf{z}^*_\eta} - \mathbf{w}_\eta^\top \mathbf{r}_j]_2, [\mathbf{w}_\eta^\top \mathbf{r}_j]_2, [\mathbf{r}_j]_2 \Big\}_{j\in[Q]} \end{aligned} \tag{26}$$

where $\mathbf{w}_\eta, \mathbf{r}_j \leftarrow \mathbb{Z}_p^k$ for all $j \in [Q]$.

| **Game** | $\mathsf{ct}^*$ | | | | $\mathsf{sk}_f$ | | **Remark** |
|---|---|---|---|---|---|---|---|
| | $\mathsf{ct}_1$ | $\mathsf{ct}_i, 1 < i < \eta$ | $\mathsf{ct}_\eta$ | $\mathsf{ct}_i, \eta < i \leq N$ | $\mathsf{sk}_{f,1}$ | $\mathsf{sk}_{f,2}$ | |
| 0 | real: $\mathbf{x}_1^*, \mathbf{z}_1^* \| - \sum \mathbf{w}_i$ | real: $\mathbf{x}_i^*, \mathbf{z}_i^* \| \mathbf{w}_i$ | | | real: | real: | Real game |
| 1 | sim : $\mathbf{x}_1^*$ | real: $\mathbf{x}_i^*, \mathbf{z}_i^* \| \mathbf{w}_i$ | | | sim : $[f(\mathbf{x}_1^*)^\top \mathbf{z}_1^* - \sum \mathbf{w}_i^\top \mathbf{r}]_2$ | real: | $\Pi_{\mathsf{ext}}$ |
| $\eta.0$ | sim: $\mathbf{x}_1^*$ | real:$\mathbf{x}_i^*, \mathbf{0} \| \mathbf{w}_i$ | real: $\mathbf{x}_\eta^*, \mathbf{z}_\eta^* \| \mathbf{w}_\eta$ | real:$\mathbf{x}_i^*, \mathbf{z}_i^* \| \mathbf{w}_i$ | sim: $[\sum_{i<\eta} f(\mathbf{x}_i^*)^\top \mathbf{z}_i^* - \sum \mathbf{w}_i^\top \mathbf{r}]_2$ | real: | |
| $\eta.1$ | sim: $\mathbf{x}_1^*$ | real:$\mathbf{x}_i^*, \mathbf{0} \| \mathbf{w}_i$ | sim :$\mathbf{x}_\eta^*$ | real:$\mathbf{x}_i^*, \mathbf{z}_i^* \| \mathbf{w}_i$ | sim: $[\sum_{i<\eta} f(\mathbf{x}_i^*)^\top \mathbf{z}_i^* - \sum \mathbf{w}_i^\top \mathbf{r}]_2$ | sim : $[f(\mathbf{x}_\eta^*)^\top \mathbf{z}_\eta^* + \mathbf{w}_\eta^\top \mathbf{r}]_2$ | $\Pi_{\mathsf{ext}}$ |
| $\eta.2$ | sim: $\mathbf{x}_1^*$ | real:$\mathbf{x}_i^*, \mathbf{0} \| \mathbf{w}_i$ | sim: $\mathbf{x}_\eta^*$ | real:$\mathbf{x}_i^*, \mathbf{z}_i^* \| \mathbf{w}_i$ | sim: $[\sum_{i\leq\eta} f(\mathbf{x}_i^*)^\top \mathbf{z}_i^* - \sum \mathbf{w}_i^\top \mathbf{r}]_2$ | sim: $[\mathbf{w}_\eta^\top \mathbf{r}]_2$ | MDDH |
| $\eta.3$ | sim: $\mathbf{x}_1^*$ | real:$\mathbf{x}_i^*, \mathbf{0} \| \mathbf{w}_i$ | real :$\mathbf{x}_\eta^*, \mathbf{0} \| \mathbf{w}_\eta$ | real:$\mathbf{x}_i^*, \mathbf{z}_i^* \| \mathbf{w}_i$ | sim: $[\sum_{i\leq\eta} f(\mathbf{x}_i^*)^\top \mathbf{z}_i^* - \sum \mathbf{w}_i^\top \mathbf{r}]_2$ | real : | $\Pi_{\mathsf{ext}}$ |
| $N.3$ | sim: $\mathbf{x}_1^*$ | real: $\mathbf{x}_i^*, \mathbf{0} \| \mathbf{w}_i$ | | | sim: $[\sum_{i\in[N]} f(\mathbf{x}_i^*)^\top \mathbf{z}_i^* - \sum \mathbf{w}_i^\top \mathbf{r}]_2$ | real : | Simulator |

**Fig. 4.** Game sequence for $\Pi_{\mathsf{ubd}}$ with $\eta \in [2, N]$, where $\mathsf{Game}_{2.0} = \mathsf{Game}_1, \mathsf{Game}_{3.0} = \mathsf{Game}_{2,3}, \ldots, \mathsf{Game}_{N,0} = \mathsf{Game}_{N-1,3}$.Each cell is in the format "xxx:yyy" where xxx $\in \{\mathsf{real}, \mathsf{sim}\}$ indicates whether the ciphertext/key component is generated using real algorithm or simulator and yyy gives out the information fed to algorithm/simulator. Throughout, the first input to $\mathsf{KeyGen}_{\mathsf{ext}}$/ $\mathsf{KeyGen}^*_{\mathsf{ext}}$ for generating $\mathsf{sk}_{f,1}$ is $(f, [\mathbf{r}]_2)$; the same applies to $\mathsf{sk}_{f,2}$. The sum of $\mathbf{w}_i^\top \mathbf{r}$ is always over $i \in [2, N]$.

$\mathsf{Game}_{\eta.3}$ for $\eta \in [2, N]$: Identical to $\mathsf{Game}_{\eta.2}$ except for the boxed terms below:

- we generate $\mathsf{mpk} = (\mathsf{mpk}_1, \mathsf{mpk}_2)$ and $\mathsf{msk} = (\mathsf{msk}_1^*, \boxed{\mathsf{msk}_2})$ where

$$(\mathsf{mpk}_1, \mathsf{msk}_1^*) \leftarrow \mathsf{Setup}_{\mathsf{ext}}^*(1^\lambda, 1^n, 1^{n'}), \boxed{(\mathsf{mpk}_2, \mathsf{msk}_2) \leftarrow \mathsf{Setup}_{\mathsf{ext}}(1^\lambda, 1^n, 1^{n'})}$$

- the challenge ciphertext for $(\mathbf{x}_1^*, \mathbf{z}_1^*, \ldots, \mathbf{x}_N^*, \mathbf{z}_N^*)$ is $\mathsf{ct}^* = (\mathsf{ct}_1^*, \mathsf{ct}_2, \ldots, \mathsf{ct}_{\eta-1}, \boxed{\mathsf{ct}_\eta}, \mathsf{ct}_{\eta+1}, \ldots, \mathsf{ct}_N)$ where

$$\mathsf{ct}_1^* \leftarrow \mathsf{Enc}_{\mathsf{ext}}^*(\mathsf{msk}_1^*, \mathbf{x}_1^*), \begin{cases} \mathsf{ct}_i \leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}_i^*, \mathbf{0} \| \mathbf{w}_i)) & i \in [2, \eta-1] \\ \boxed{\mathsf{ct}_i \leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}_\eta^*, \mathbf{0} \| \mathbf{w}_\eta))} & i = \eta \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}_i^*, \mathbf{z}_i^* \| \mathbf{w}_i)) & i \in [\eta+1, N] \end{cases}$$

- the key for the $j$-th query $f_j$ is $\mathsf{sk}_{f_j} = (\mathsf{sk}_{f_j,1}^*, \boxed{\mathsf{sk}_{f_j,2}}, [\mathbf{r}_j]_2)$ where

$$\mathsf{sk}_{f_j,1}^* \leftarrow \mathsf{KeyGen}_{\mathsf{ext}}^*\big(\mathsf{msk}_1^*, \mathbf{x}_1^*, (f_j, [\mathbf{r}_j]_2), [\textstyle\sum_{i\in[\eta]} f_j(\mathbf{x}_i^*)^\top \mathbf{z}_i^* - \sum_{i\in[2,N]} \mathbf{w}_i^\top \mathbf{r}_j]_2\big)$$

$$\boxed{\mathsf{sk}_{f_j,2} \leftarrow \mathsf{KeyGen}_{\mathsf{ext}}(\mathsf{msk}_2, (f_j, [\mathbf{r}_j]_2))}$$

where $\mathbf{w}_2, \ldots, \mathbf{w}_N \leftarrow \mathbb{Z}_p^k$ and $\mathbf{r}_j \leftarrow \mathbb{Z}_p^k$ for all $j \in [Q]$. We claim that $\mathsf{Game}_{\eta.2} \approx_c \mathsf{Game}_{\eta.3}$. This follows from the simulation-based semi-adaptive security of $\Pi_{\mathsf{ext}}$ with the fact $f_j(\mathbf{x}_\eta^*)^\top \mathbf{0} + \mathbf{w}_\eta^\top \mathbf{r} = \mathbf{w}_\eta^\top \mathbf{r}$.

Here we have $\mathsf{Game}_{2.0} = \mathsf{Game}_1$ and $\mathsf{Game}_{\eta.0} = \mathsf{Game}_{\eta-1.3}$ for all $\eta \in [3, N]$. Note that $\mathsf{Game}_{N.3}$ corresponds to the output of the simulator in the ideal game. We summarize the game sequence in Fig. 4. We prove the indistinguishability of adjacent games listed above in the full paper.

## References


1. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 733–751. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46447-2_33
2. Abdalla, M., Catalano, D., Gay, R., Ursu, B.: Inner-product functional encryption with fine-grained access control. Cryptology ePrint Archive, Report 2020/577 (2020)
3. Abdalla, M., Gay, R., Raykova, M., Wee, H.: Multi-input inner-product functional encryption from pairings. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 601–626. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_21
4. Agrawal, S., Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption: new perspectives and lower bounds. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 500–518. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_28
5. Agrawal, S., Libert, B., Maitra, M., Titiu, R.: Adaptive simulation security for inner product functional encryption. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part I. LNCS, vol. 12110, pp. 34–64. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45374-9_2

6. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 333–362. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_12
7. Agrawal, S., Maitra, M., Yamada, S.: Attribute based encryption (and more) for nondeterministic finite automata from LWE. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 765–797. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26951-7_26
8. Agrawal, S., Maitra, M., Yamada, S.: Attribute based encryption for deterministic finite automata from DLIN. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019, Part II. LNCS, vol. 11892, pp. 91–117. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36033-7_4
9. Ambrona, M., Barthe, G., Gay, R., Wee, H.: Attribute-based encryption in the generic group model: automated proofs and new constructions. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017, pp. 647–664. ACM Press, October/November 2017
10. Baltico, C.E.Z., Catalano, D., Fiore, D., Gay, R.: Practical functional encryption for quadratic functions with applications to predicate encryption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 67–98. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_3
11. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: 20th ACM STOC, pp. 1–10. ACM Press, May 1988
12. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit ABE and Compact Garbled Circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 533–556. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_30
13. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_16
14. Brakerski, Z., Vaikuntanathan, V.: Circuit-ABE from LWE: unbounded attributes and semi-adaptive security. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 363–384. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_13
15. Chen, J., Gay, R., Wee, H.: Improved dual system ABE in prime-order groups via predicate encodings. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 595–624. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_20
16. Chen, J., Wee, H.: Semi-adaptive attribute-based encryption and improved delegation for Boolean formula. In: Abdalla, M., De Prisco, R. (eds.) SCN 2014. LNCS, vol. 8642, pp. 277–297. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10879-7_16
17. Chen, Y., Zhang, L., Yiu, S.-M.: Practical attribute based inner product functional encryption from simple assumptions. Cryptology ePrint Archive, Report 2019/846 (2019)
18. Chotard, J., Dufour Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Decentralized multi-client functional encryption for inner product. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 703–732. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03329-3_24

19. Datta, P., Okamoto, T., Takashima, K.: Adaptively simulation-secure attribute-hiding predicate encryption. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 640–672. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03329-3_22
20. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for Diffie-Hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 129–147. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_8
21. Goldwasser, S., Gordon, S.D., Goyal, V., Jain, A., Katz, J., Liu, F.-H., Sahai, A., Shi, E., Zhou, H.-S.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 578–602. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_32
22. Gong, J., Waters, B., Wee, H.: ABE for DFA from $k$-Lin. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11693, pp. 732–764. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26951-7_25
23. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC, pp. 545–554. ACM Press, June 2013
24. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Predicate encryption for circuits from LWE. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 503–523. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_25
25. Goyal, R., Koppula, V., Waters, B.: Semi-adaptive security and bundling functionalities made generic and easy. In: Hirt, M., Smith, A. (eds.) TCC 2016, Part II. LNCS, vol. 9986, pp. 361–388. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_14
26. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., De Capitani di Vimercati, S. (eds.) ACM CCS 2006, pp. 89–98. ACM Press, October/November 2006. Available as Cryptology ePrint Archive Report 2006/309
27. Ishai, Y., Wee, H.: Partial garbling schemes and their applications. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) ICALP 2014, Part I. LNCS, vol. 8572, pp. 650–662. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43948-7_54
28. Jain, A., Lin, H., Matt, C., Sahai, A.: How to leverage hardness of constant-degree expanding polynomials over $\mathbb{R}$ to build $i\mathcal{O}$. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 251–281. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_9
29. Jain, A., Lin, H., Sahai, A.: Simplifying constructions and assumptions for $i\mathcal{O}$. IACR Cryptology ePrint Archive, 2019:1252 (2019)
30. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_9
31. Kowalczyk, L., Wee, H.: Compact adaptively secure ABE for $\mathsf{NC}^1$ from $k$-Lin. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 3–33. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_1
32. Lewko, A., Waters, B.: Unbounded HIBE and attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 547–567. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_30

33. Lin, H.: Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 599–629. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_20
34. Okamoto, T., Takashima, K.: Adaptively attribute-hiding (hierarchical) inner product encryption. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 591–608. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_35
35. Okamoto, T., Takashima, K.: Efficient (hierarchical) inner-product encryption tightly reduced from the decisional linear assumption. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **96**–**A**(1), 42–52 (2013)
36. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_27
37. Waters, B.: Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_36
38. Waters, B.: Functional encryption for regular languages. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 218–235. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_14
39. Wee, H.: Dual system encryption via predicate encodings. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 616–637. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54242-8_26
40. Wee, H.: Attribute-hiding predicate encryption in bilinear groups, revisited. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 206–233. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_8