Functional Encryption for Attribute-Weighted Sums from k-Lin

Michel Abdalla, Junqing Gong, Hoeteck Wee

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
michel.abdalla@ens.fr, wee@di.ens.fr
[2] East China Normal University
jqgong@sei.ecnu.edu.cn
[3] NTT Research

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.

— Contents —

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

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

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).

⋆ Supported by ERC Project aSCEND (H2020 639554) and the French FUI project ANBLIC.
⋆⋆ 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.
⋆⋆⋆ Supported in part by ERC Project aSCEND (H2020 639554).

This work. We study functional encryption (FE) schemes for attribute-weighted sums [39,28,15,26], 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 ct, and generate secret keys sk_f so that decrypting ct with 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. As articulated [15], simulation-based security is the right notion for functional encryption applied to real-world data.

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 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 [41,9]. In an unbounded-slot FE, key generation and the size of 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 [35,16,27,21]), where each 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[4], simulation-based security against unbounded collusions;
(4) relies on the standard k-linear assumption in prime-order bilinear groups.

As with all prior FE schemes that rely on DDH and bilinear groups [3,8,5,36,12,31,32,19], 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 [23,20], 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 [3,8] where f outputs a fixed vector,
(ii) attribute-based encryption (ABE) by taking z to be the payload,
(iii) attribute-based inner-product FE [19,4], 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 [31,32] do capture a larger class supporting quadratic instead of linear functions over $\mathbf{z}$,[5] but in a weaker secret-key setting, which is nonetheless sufficient for the application to obfuscation. Finally, none of these works consider the unbounded-slot setting.

[4] We actually achieve semi-adaptive security [18], a slight strengthening of selective security.
[5] Note that we can also capture the same class with a quadratic blow-up in ciphertext size.

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 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 [43] (which in turn builds on [33,40,42,30]) 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 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 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 [6], (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

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

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 [24], notably the idea of propagating entropy along the execution path of a DFA.

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

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

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

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

  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 $N = 2$:

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

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 mpk_1 and the remaining $\mathsf{ct}_{\mathbf{x}_i,\mathbf{z}_i\|w_i}, i = 2,\ldots,N$ w.r.t. 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.

Scheme	Enc	KeyGen	**Function**	**Security**	$\lvert\mathsf{ct}\rvert$
OT12, KSW08 [37,38,33]	$\mathbf{z}$	$\mathbf{y}$	$\mathbf{z}^\top\mathbf{y} \stackrel{?}{=} 0$	AD-IND	$O(\lvert\mathbf{z}\rvert)$
ALS16, ABDP15 [3,8]	$\mathbf{z}$	$\mathbf{y}$	$\mathbf{z}^\top\mathbf{y}$	AD-IND	$O(\lvert\mathbf{z}\rvert)$
W17 [43]	$\mathbf{x},\mathbf{z}$	f ABP	$\mathbf{z}^\top f(\mathbf{x}) \stackrel{?}{=} 0$	SA-SIM	$O(\lvert\mathbf{x}\rvert+\lvert\mathbf{z}\rvert)$
DOT18 [21]	$\mathbf{x},\mathbf{z}$	f ABP	$\mathbf{z}^\top f(\mathbf{x}) \stackrel{?}{=} 0$	AD-SIM	$O(\lvert\mathbf{x}\rvert+\lvert\mathbf{z}\rvert)$
ACGU20, CZY19 [4,19]	$\mathbf{x},\mathbf{z}$	$\mathbf{y}, f$ NC1	$f(\mathbf{x})\cdot\mathbf{z}^\top\mathbf{y}$	AD-IND	$O(\lvert\mathbf{x}\rvert+\lvert\mathbf{z}\rvert)$
ACGU20 [4]	$\mathbf{z}_1,\mathbf{z}_2$	$\mathbf{y}_1,\mathbf{y}_2$	$\mathbf{z}_1^\top\mathbf{y}_1$ 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 $\mathbf{x}$.

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. mpk_2 in the presence of a simulated secret key w.r.t. mpk_1; the second goes away because the two simulated ciphertext correspond to mpk_1 and mpk_2 respectively. We defer the remaining details to the technical overview in Section 2 and the formal scheme in Section 7.

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 [23,20]; 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 [13]. 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 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 mpk_1 and another of $(\mathbf{x}_i,\mathbf{z}-\mathbf{z}_i'\|w_i-w_i')$ under mpk_2. This readily gives us a multi-client unbounded-slot FE for attribute-weighted sums; we refer the reader to Section 8 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 [41,24,9,10]. At a very high level, our construction may be viewed as following the paradigm in [9,10] 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 [24], 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

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

Fig. 2. Summary of ciphertext and key sizes of our one-slot scheme Π_{one} and unbounded-slot scheme Π_{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.

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 [11]?
- 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 [21,7], and more recent advances in [34] 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 [25,14,8]; 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 Section 2. We present preliminaries, definitions and tools in Sections 3 and 4. We present our one-slot scheme and an extension in Sections 5 and 6, and the unbounded-slot scheme and the multi-client extension in Sections 7 and 8.

2 Technical Overview

We proceed with a more technical overview of our construction, building on the overview given in Section 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

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

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 $\ell_1(\mathbf{x}), \ldots, \ell_m(\mathbf{x})$,[6] 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 $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ [22]. Our starting point is the following scheme[7]:

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

where

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

Decryption uses the fact that

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

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 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

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

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

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

[7] The scheme in [43] has a larger ciphertext of the form: $\mathsf{ct}_{\mathbf{x},\mathbf{z}} = \left([s]_1, [\mathbf{z} + s\mathbf{w}]_1, [s(\mathbf{u} + v\mathbf{x})]_1\right) \in \mathbb{G}_1^{n+n'+1}$.

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

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

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[8]

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

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

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

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 $\mathbf{u} \leftarrow \mathbb{Z}_p^{kn}, \mathbf{R} \leftarrow \mathbb{Z}_p^{k\times m}$.

The compiler. To obtain a public-key scheme secure under the k-Lin assumption, we perform the following substitutions to (6), following [43,17]:

$$\begin{aligned} s &\mapsto \mathbf{s}^\top\mathbf{A}^\top \in \mathbb{Z}_p^{1\times(k+1)} \\ \mathbf{w}^\top &\mapsto \mathbf{W} \in \mathbb{Z}_p^{(k+1)\times n'} \\ \mathbf{u}^\top &\mapsto \mathbf{U} \in \mathbb{Z}_p^{(k+1)\times kn} \\ v &\mapsto \mathbf{V} \in \mathbb{Z}_p^{(k+1)\times k} \\ \mathbf{t}^\top &\mapsto \mathbf{T} \in \mathbb{Z}_p^{(k+1)\times(m+n'-1)} \\ \mathbf{r}^\top &\mapsto \mathbf{R} \in \mathbb{Z}_p^{k\times m}\end{aligned}$$

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 [43]:

- We first switch $[\mathbf{s}^\top\mathbf{A}^\top]_1$ in the ciphertext with a random $[\mathbf{c}^\top]_1$.
- We decompose 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 (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 Section 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 Section 1.2 for a high-level overview of the unbounded-slot scheme, and proceed directly to describe the construction and the security proof.

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

$$
\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} & \boxed{\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) \\
& \boxed{\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 \boxed{\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), \boxed{\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 + \boxed{f(\mathbf{x}_2)^\top \mathbf{z}_2} - w_2 r - w_3 r]_2), \mathsf{KeyGen}_2^*((f, [r]_2), [\boxed{w_2 r}]_2) \\
\overset{\text{SIM-2}}{\approx_c} & \mathsf{Enc}_1^*(\mathbf{x}_1), \quad \boxed{\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), \boxed{\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 \boxed{\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), \boxed{\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 + \boxed{f(\mathbf{x}_3)^\top \mathbf{z}_3} - w_2 r - w_3 r]_2), \mathsf{KeyGen}_2^*((f, [r]_2), [\boxed{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 \boxed{\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), \boxed{\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 construction. We run two copies of the one-slot scheme, which we denote by

$$(\mathsf{Enc}_b, \mathsf{KeyGen}_b) = (\mathsf{Enc}(\mathsf{mpk}_b, \cdot), \mathsf{KeyGen}(\mathsf{msk}_b, \cdot)), \ b = 1, 2$$

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

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

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

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

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 ϕ. An arithmetic branching program (ABP), where p is a prime, computes a function $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$. Here, ϕ 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^λ 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 λ. 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 $\mathbb{G}_1$ [22]:

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

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

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.* [22] 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 $\text{MDDH}^1_{k,Q}$.

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 $2 \cdot \mathsf{Adv}_{\mathcal{B}}^{\mathrm{MDDH}^1_{k,Q}}(\lambda)$.

Proof. This follows from $\mathrm{MDDH}^1_{k,Q}$ assumption stating that $\{[\mathbf{w}^\top \mathbf{r}_j]_2, [\mathbf{r}_j]\}_j \approx \{[\hat{r}_j]_2, [\mathbf{r}_j]_2\}_j$ with $\hat{r}_j \leftarrow \mathbb{Z}_p$ for all $j \in [Q]$ and change of variables: $\hat{r}_j \mapsto \hat{r}_j - \mu_j$ or all $j \in [Q]$. □

4 Definitions and Tools

In this section, we formalize functional encryption for attribute-weighted sums, using the framework of partially-hiding functional encryption [26,43,15].

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^λ and function parameters $1^n, 1^{n'}$. It outputs the master public key mpk and the master secret key msk.

$\mathsf{Enc}(\mathsf{mpk}, (\mathbf{x}_i, \mathbf{z}_i)_{i\in[N]})$: The encryption algorithm gets as input 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 msk and a function $f \in \mathcal{F}_{\mathsf{ABP},n,n'}$. It outputs a secret key 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 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

$$\Pr\left[\mathsf{Dec}((\mathsf{ct}_{(\mathbf{x}_i,\mathbf{z}_i)}, (\mathbf{x}_i)_{i\in[N]}), (\mathsf{sk}_f, f)) = \textstyle\sum_{i\in[N]} f(\mathbf{x}_i)^\top \mathbf{z}_i \ : \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^{n'}) \\ \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f) \\ \mathsf{ct}_{(\mathbf{x}_i,\mathbf{z}_i)} \leftarrow \mathsf{Enc}(\mathsf{mpk}, (\mathbf{x}_i, \mathbf{z}_i)_{i\in[N]}) \end{array} \right] = 1.$$

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 [3,8,5,36,12], 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 [18] (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 KeyGen, the simulator 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 [30,43] 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:

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

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 $(\mathbf{p}_{f,\mathbf{x},\mathbf{z}}^\top, f, \mathbf{x})$, we can recover $f(\mathbf{x})^\top \mathbf{z}$, while learning nothing else about $\mathbf{z}$.

Lemma 2 (partial garbling [30,43]). *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* $f,\mathbf{x},\mathbf{z},\mathbf{t}$,

$$\mathbf{p}_{f,\mathbf{x},\mathbf{z}}^\top \mathbf{d}_{f,\mathbf{x}} = f(\mathbf{x})^\top \mathbf{z}$$

  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 Section A for more detail about Lemma 2 and the extension.

5 Π_{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 Π_{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. Section 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. Section 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 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) $$\mathbf{z}^\top - \mathbf{s}^\top\mathbf{A}^\top\underline{\mathbf{T}} = (\mathbf{z}^\top + \mathbf{s}^\top\mathbf{A}^\top\mathbf{W}) \cdot \mathbf{I}_{n'} - \mathbf{s}^\top\mathbf{A}^\top \cdot (\underline{\mathbf{T}} + \mathbf{W})$$

(13) $$\mathbf{s}^\top\mathbf{A}^\top\mathbf{T}\mathbf{L}_1(\mathbf{x}\otimes\mathbf{I}_m) + \mathbf{s}^\top\mathbf{A}^\top\mathbf{T}\mathbf{L}_0 = \mathbf{s}^\top\mathbf{A}^\top \cdot \left((\mathbf{T}\mathbf{L}_1 + \mathbf{U}(\mathbf{I}_n\otimes\mathbf{R}))(\mathbf{x}\otimes\mathbf{I}_m) + (\mathbf{T}\mathbf{L}_0 + \mathbf{V}\mathbf{R})\right) - \left(\mathbf{s}^\top\mathbf{A}^\top\mathbf{U}(\mathbf{x}\otimes\mathbf{I}_k) + \mathbf{s}^\top\mathbf{A}^\top\mathbf{V}\right) \cdot \mathbf{R}$$

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 [43]). The ciphertext in [43] 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 [43] are $O(mn + n')$. In our scheme, the multiplicative factor of n comes at the cost of a smaller ciphertext. In [43], 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 Π_{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

$$\mathsf{mpk} = \left(\mathbb{G}, [\mathbf{A}^\top]_1, [\mathbf{A}^\top \mathbf{W}]_1, [\mathbf{A}^\top \mathbf{U}]_1, [\mathbf{A}^\top \mathbf{V}]_1\right) \quad \text{and} \quad \mathsf{msk}^* = \left(\mathbf{W}, \mathbf{U}, \mathbf{V}, \widetilde{\mathbf{w}}, \widetilde{\mathbf{v}}, \mathbf{c}, \mathbf{C}^\perp, \mathbf{A}, \mathbf{a}^\perp\right)$$

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

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

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

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

where

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

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

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

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 $(\mathbf{p}_1^\top, \mathbf{p}_2^\top)\mathbf{d}_{f,\mathbf{x}} = f(\mathbf{x})^\top \mathbf{z}$ where $(\mathbf{p}_1^\top, \mathbf{p}_2^\top) = \mathsf{pgb}(f, \mathbf{x}, \mathbf{z}; (\mathbf{s}^\top \mathbf{A}^\top \mathbf{T})^\top)$ as in the real 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*

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

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.

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

$\underline{\mathsf{Game}_1}$: Identical to Game_0 except that 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 mpk and $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ along with $[\mathbf{A}^\top]_1$; the challenge ciphertext ct^* is generated using the challenge term given above. See Lemma 4 for details.

$\underline{\mathsf{Game}_2}$: Identical to Game_1 except that the j-th query f_j to $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ 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

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

where $(\mathbf{L}_{1,j}, \mathbf{L}_{0,j}) \leftarrow \mathsf{lgen}(f_j)$, $\mathbf{T}_j \leftarrow \mathbb{Z}_p^{(k+1)\times(m+n'-1)}$, $\mathbf{R}_j \leftarrow \mathbb{Z}_p^{k\times m}$, $\mathbf{c}$ is the randomness in 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. Setup^* in Section 5.2). By basic linear algebra, we have $\mathsf{Game}_1 = \mathsf{Game}_2$.

$\underline{\mathsf{Game}_3}$: Identical to Game_2 except that we replace $\mathsf{Setup}, \mathsf{Enc}$ with $\mathsf{Setup}^*, \mathsf{Enc}^*$ where 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 Game_2 except that, for the j-th query f_j, we compute

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

where $\tilde{\mathbf{w}}, \tilde{\mathbf{v}}$ are given in msk^* (output by 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

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

which is implied by Lemma 3. See Lemma 5 for details.

$\underline{\mathsf{Game}_4}$: Identical to Game_3 except that we replace KeyGen_3^* with KeyGen_4^* which works as KeyGen_3^* except that, for the j-th query f_j, we compute

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

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. See Lemma 6 for details.

$\underline{\mathsf{Game}_5}$: Identical to Game_4 except that we replace KeyGen_4^* with 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 Section 4.2. See Lemma 7 for details.

We use $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{xx}}(\lambda)$ to denote the advantage of adversary $\mathcal{A}$ in $\mathsf{Game}_{\mathsf{xx}}$.

5.4 Lemmas

We prove the following lemmas showing the indistinguishability of adjacent games listed above. This completes the proof of Theorem 2.

Lemma 4 ($\mathsf{Game}_0 \approx_c \mathsf{Game}_1$)**.** *For all $\mathcal{A}$, there exists $\mathcal{B}_1$ with $\mathsf{Time}(\mathcal{B}_1) \approx \mathsf{Time}(\mathcal{A})$ such that*

$$|\mathsf{Adv}_{\mathcal{A}}^1(\lambda) - \mathsf{Adv}_{\mathcal{A}}^0(\lambda)| \leq \mathsf{Adv}_{\mathcal{B}_1}^{\mathrm{MDDH}_{k,k+1}^1}(\lambda).$$

Proof. Recall that we replace $\boxed{\mathbf{s}^\top \mathbf{A}^\top}$ with $\mathbf{c}^\top \leftarrow \mathbb{Z}_p^{1 \times (k+1)}$ in ct^* in Game_1. We prove the lemma from $\mathrm{MDDH}_{k,k+1}^1$ assumption:

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

where $\mathbf{s} \leftarrow \mathbb{Z}_p^k$ and $\mathbf{c} \leftarrow \mathbb{Z}_p^{k+1}$. On input $[\mathbf{A}^\top]_1, [\mathbf{c}^\top]_1$ where

$$\mathbf{c}^\top = \boxed{\mathbf{s}^\top \mathbf{A}^\top} \quad \text{or} \quad \mathbf{c}^\top \leftarrow \mathbb{Z}_p^{1 \times (k+1)},$$

the algorithm $\mathcal{B}_1$ samples $\mathsf{msk} = (\mathbf{W}, \mathbf{U}, \mathbf{V})$ and proceeds as follows:

- simulate mpk using msk and $[\mathbf{A}^\top]_1$;

– on input the semi-adaptive challenge $(\mathbf{x}^*, \mathbf{z}^*)$, output the challenge ciphertext

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

using $[\mathbf{c}]_1$ and msk;

– we answer all key queries honestly using msk.

Observe that, when $\mathbf{c}^\top = \boxed{\mathbf{s}^\top \mathbf{A}^\top}$, the simulation is identical to Game_0; when $\mathbf{c}^\top \leftarrow \boxed{\mathbb{Z}_p^{1\times(k+1)}}$, the simulation is identical to Game_1. This proves the lemma. □

Lemma 5 ($\mathsf{Game}_2 \approx_c \mathsf{Game}_3$)**.** *For all $\mathcal{A}$, we have $\mathsf{Adv}^3_{\mathcal{A}}(\lambda) \approx \mathsf{Adv}^2_{\mathcal{A}}(\lambda)$.*

Proof (of Lemma 5). Recall that the difference between the two games lies in ct^* and $\mathsf{sk}_{f_j}[2]$: instead of computing

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

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

in Game_2, we compute

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

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

in Game_3.

This follows readily from the following statement, which in turn follows from Lemma 3: for all $\mathbf{x}^*, \mathbf{z}^*$,

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

where $\mathbf{U}, \mathbf{W}, \mathbf{V}, \tilde{\mathbf{w}}, \tilde{\mathbf{v}}$ are sampled as in Setup^* and $\tilde{\mathbf{u}} \leftarrow \mathbb{Z}_p^{kn}, \mathbf{T}_j \leftarrow \mathbb{Z}_p^{(k+1)\times(m+n'-1)}, \tilde{\mathbf{t}}_j \leftarrow \mathbb{Z}_p^{m+n'-1}$. We clarify that in the semi-adaptive security game, $(\mathbf{x}^*, \mathbf{z}^*)$ are chosen *after* seeing $\mathbf{A}^\top \mathbf{U}, \mathbf{A}^\top \mathbf{W}, \mathbf{A}^\top \mathbf{V}$. Since the two distributions are identically distributed, the distinguishing advantage remains 0 even for adaptive choices of $\mathbf{x}^*, \mathbf{z}^*$ via a random guessing argument.

Finally, note that $\mathbf{A}^\top \mathbf{U}, \mathbf{A}^\top \mathbf{W}, \mathbf{A}^\top \mathbf{V}, \mathbf{A}^\top \mathbf{T}_j$ are used to simulate $\mathsf{mpk}, \mathsf{sk}_{f_j}[1]$, whereas the boxed/gray terms are used to simulate $\mathsf{sk}_{f_j}[2]$. This readily proves the lemma. □

Lemma 6 ($\mathsf{Game}_3 \approx_c \mathsf{Game}_4$)**.** *For all $\mathcal{A}$, there exists $\mathcal{B}_2$ with $\mathsf{Time}(\mathcal{B}_2) \approx \mathsf{Time}(\mathcal{A})$ such that*

$$|\mathsf{Adv}^4_{\mathcal{A}}(\lambda) - \mathsf{Adv}^3_{\mathcal{A}}(\lambda)| \le \mathsf{Adv}^{\mathrm{MDDH}^n_{k,mQ}}_{\mathcal{B}_2}(\lambda)$$

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

Proof. Recall that the differences between the two games is that we replace $\boxed{\tilde{\mathbf{u}}^\top (\mathbf{I}_n \otimes \mathbf{R}_j)}$ with $\boxed{\hat{\mathbf{u}}_j^\top}$ in $\mathsf{sk}_{f_j}[2]$ for all $j \in [Q]$. We prove the lemma from $\mathrm{MDDH}^n_{k,mQ}$ assumption which implies that

$$\left\{ \boxed{[\tilde{\mathbf{u}}^\top (\mathbf{I}_n \otimes \mathbf{R}_j)]_2}, [\mathbf{R}_j]_2 \right\}_{j\in[Q]} \approx_c \left\{ \boxed{[\hat{\mathbf{u}}_j^\top]_2}, [\mathbf{R}_j]_2 \right\}_{j\in[Q]} \tag{16}$$

where $\tilde{\mathbf{u}} \leftarrow \mathbb{Z}_p^{kn}, \mathbf{R}_j \leftarrow \mathbb{Z}_p^{k\times m}, \hat{\mathbf{u}}_j \leftarrow \mathbb{Z}_p^{nm}$. To see that $\mathrm{MDDH}^n_{k,mQ}$ implies (16), we write

$$\tilde{\mathbf{u}}^\top = (\tilde{\mathbf{u}}_1^\top, \ldots, \tilde{\mathbf{u}}_n^\top) \in (\mathbb{Z}_p^{1\times k})^n \quad \text{and} \quad \hat{\mathbf{u}}_j^\top = (\hat{\mathbf{u}}_{j,1}^\top, \ldots, \tilde{\mathbf{u}}_{j,n}^\top) \in (\mathbb{Z}_p^{1\times m})^n, \forall\, j \in [Q]$$

and restate (16) as follows:

$$[\mathbf{MS}]_2, [\mathbf{M}]_2 \approx_c [\mathbf{U}]_2, [\mathbf{M}]_2$$

where

$$\mathbf{S} = (\tilde{\mathbf{u}}_1 | \cdots | \tilde{\mathbf{u}}_n) \leftarrow \mathbb{Z}_p^{k \times n},\ \mathbf{M} = \begin{pmatrix} \mathbf{R}_1^\top \\ \vdots \\ \mathbf{R}_Q^\top \end{pmatrix} \leftarrow \mathbb{Z}_p^{mQ \times k},\ \mathbf{U} = \begin{pmatrix} \hat{\mathbf{u}}_{1,1} & \cdots & \hat{\mathbf{u}}_{1,n} \\ \vdots & & \vdots \\ \hat{\mathbf{u}}_{Q,1} & \cdots & \hat{\mathbf{u}}_{Q,n} \end{pmatrix} \leftarrow \mathbb{Z}_p^{mQ \times n},$$

this is exactly that $\mathrm{MDDH}^n_{k,mQ}$ assumption states.

We proceed to prove the lemma from (16). On input $\{[\mathbf{t}_j^\top]_2, [\mathbf{R}_j]_2\}_{j \in [Q]}$ where

$$\mathbf{t}_j^\top = \boxed{\tilde{\mathbf{u}}^\top (\mathbf{I}_n \otimes \mathbf{R}_j)} \quad \text{or} \quad \mathbf{t}_j^\top = \hat{\mathbf{u}}_j^\top,$$

the algorithm $\mathcal{B}$ works as follows:

Setup. Run Setup^* honestly with output

$$\mathsf{mpk} = \left(\mathbb{G}, [\mathbf{A}^\top]_1, [\mathbf{A}^\top \mathbf{W}]_1, [\mathbf{A}^\top \mathbf{U}]_1, [\mathbf{A}^\top \mathbf{V}]_1\right) \quad \text{and} \quad \mathsf{msk}^* = \left(\mathbf{W}, \mathbf{U}, \mathbf{V}, \tilde{\mathbf{w}}, \tilde{\mathbf{v}}, \mathbf{c}, \mathbf{C}^\perp, \mathbf{A}, \mathbf{a}^\perp\right)$$

and return mpk to $\mathcal{A}$.

Challenge. On the semi-adaptive challenge $(\mathbf{x}^*, \mathbf{z}^*)$, output $\mathsf{Enc}^*(\mathsf{msk}^*, \mathbf{x}^*)$.

Key queries. On input the j-th key query f_j, we sample $\mathbf{T}_j \leftarrow \mathbb{Z}_p^{(k+1)\times(m+n'-1)}, \tilde{\mathbf{t}}_j \leftarrow \mathbb{Z}_p^{m+n'-1}$ and want to return sk_{f_j} combined from the components below via (15):

$$\begin{aligned} \mathsf{sk}_{f_j}[1] &= \left([\mathbf{A}^\top \underline{\mathbf{T}}_j + \mathbf{A}^\top \mathbf{W}]_2, [\mathbf{A}^\top \mathbf{T}_j \mathbf{L}_{1,j} + \mathbf{A}^\top \mathbf{U}(\mathbf{I}_n \otimes \mathbf{R}_j)]_2, [\mathbf{A}^\top \mathbf{T}_j \mathbf{L}_{0,j} + \mathbf{A}^\top \mathbf{V} \mathbf{R}_j]_2\right) \\ \mathsf{sk}_{f_j}[2] &= \left([\underline{\tilde{\mathbf{t}}}_j^\top - (\mathbf{z}^*)^\top + \tilde{\mathbf{w}}^\top]_2, [\tilde{\mathbf{t}}_j^\top \mathbf{L}_{1,j} + \mathbf{t}_j^\top]_2, [\tilde{\mathbf{t}}_j^\top \mathbf{L}_{0,j} - \mathbf{t}_j^\top (\mathbf{x}^* \otimes \mathbf{I}_m) + \tilde{\mathbf{v}}^\top \mathbf{R}_j]_2\right) \end{aligned}$$

where

$$\mathbf{t}_j^\top = \begin{cases} \boxed{\tilde{\mathbf{u}}^\top (\mathbf{I}_n \otimes \mathbf{R}_j)} & , \mathsf{KeyGen}_3^* \\ \hat{\mathbf{u}}_j^\top & , \mathsf{KeyGen}_4^* \end{cases}$$

Here $\mathsf{sk}_{f_j}[1]$ can be simulated using msk^* and $[\mathbf{R}_j]_2$ given out in the input; $\mathsf{sk}_{f_j}[2]$ can be simulated using msk^*, $[\mathbf{R}_j]_2$ and the challenge term $[\mathbf{t}_j^\top]_2$ in the input.

Observe that

- when $\mathbf{t}_j^\top = \boxed{\tilde{\mathbf{u}}^\top (\mathbf{I}_n \otimes \mathbf{R}_j)}$, we simulate KeyGen_3^* during **Key queries** and the simulation is identical to Game_3;
- when $\mathbf{t}_j^\top = \hat{\mathbf{u}}_j^\top$, we simulate KeyGen_4^* during **Key queries** and the simulation is identical to Game_4.

This proves the lemma. □

Lemma 7 ($\mathsf{Game}_4 \approx_s \mathsf{Game}_5$)**.** *For all $\mathcal{A}$, we have* $\mathsf{Adv}^5_{\mathcal{A}}(\lambda) \approx \mathsf{Adv}^4_{\mathcal{A}}(\lambda)$.

Proof. Recall that the difference between the two games lies in $\mathsf{sk}_{f_j}[2]$: instead of computing

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

in KeyGen_4^* (i.e., Game_4), we compute

$$\mathsf{sk}_{f_j}[2] = \left([\tilde{\mathbf{t}}_j^\top + \tilde{\mathbf{w}}^\top]_2, [\hat{\mathbf{u}}_j^\top]_2, [\tilde{\mathbf{t}}_j^\top (\mathbf{L}_{1,j}(\mathbf{x}^* \otimes \mathbf{I}_m) + \mathbf{L}_{0,j}) + \mathbf{e}_1^\top \cdot f_j(\mathbf{x}^*)^\top \mathbf{z}^* - \hat{\mathbf{u}}_j^\top (\mathbf{x}^* \otimes \mathbf{I}_m) + \tilde{\mathbf{v}}^\top \mathbf{R}]_2\right)$$

in KeyGen^* (i.e., Game_5). By change of variable $\hat{\mathbf{u}}_j^\top \mapsto \hat{\mathbf{u}}_j^\top - \tilde{\mathbf{t}}_j^\top \mathbf{L}_{1,j}$ for all $j \in [Q]$ in Game_4, we can rewrite in the form:

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

where

$$(\mathbf{p}_{j,1}^\top, \mathbf{p}_{j,2}^\top) \leftarrow \begin{cases} \boxed{\mathsf{pgb}(f_j, \mathbf{x}^*, \mathbf{z}^*; \tilde{\mathbf{t}}_j)} & \text{in } \mathsf{Game}_4 \\ \mathsf{pgb}^*(f_j, \mathbf{x}^*, f_j(\mathbf{x}^*)^\top \mathbf{z}^*; \tilde{\mathbf{t}}_j) & \text{in } \mathsf{Game}_5 \end{cases}$$

Then the lemma immediately follows from the privacy of underlying partial garbling scheme which means $\mathsf{pgb}(f_j, \mathbf{x}^*, \mathbf{z}^*) \approx_s \mathsf{pgb}^*(f_j, \mathbf{x}^*, f_j(\mathbf{x}^*)^\top \mathbf{z}^*)$. □

6 Π_{ext}: Extending Π_{one}

In this section, we extend our one-slot FE scheme Π_{one} in Section 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:

- 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 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 [8]— is to augment the previous construction Π_{one} with $[\mathbf{A}^\top\mathbf{W}_0]_1$ in 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 Π_{one}, while relying on the extension of the garbling scheme for handling shifts to argue both correctness and security, cf. Section 4.2.

6.1 Our scheme

Scheme. Our extended one-slot FE scheme Π_{ext} in prime-order bilinear group is described as follows. The boxes indicate the changes from the scheme in Section 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

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

  and output

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

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

$$\mathsf{ct}_{\mathbf{x},\mathbf{z},\mathbf{w}}=\big([\mathbf{s}^\top\mathbf{A}^\top]_1,[\mathbf{z}^\top+\mathbf{s}^\top\mathbf{A}^\top\mathbf{W}]_1,[\mathbf{s}^\top\mathbf{A}^\top\mathbf{U}(\mathbf{x}\otimes\mathbf{I}_k)+\mathbf{s}^\top\mathbf{A}^\top\mathbf{V}]_1,\boxed{[\mathbf{w}^\top+\mathbf{s}^\top\mathbf{A}^\top\mathbf{W}_0]_1}\big)\quad\text{and}\quad\mathbf{x}.$$

- $\mathsf{KeyGen}(\mathsf{msk},(f,[\mathbf{r}]_2))$: 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. Section 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[9]

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

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

[9] 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 $(f,\mathbf{r})$.

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

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

and ciphertext:

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

the decryption works as follows:

1. compute
$$[\mathbf{p}_1^\top]_T = e([\mathbf{c}_1^\top]_1, [\mathbf{I}_{n'}]_2) \cdot e([\mathbf{c}_0^\top]_1, [-\mathbf{K}_1]_2) \tag{17}$$
2. compute
$$[\mathbf{p}_2^\top]_T = e([\mathbf{c}_0^\top]_1, [\mathbf{K}_2(\mathbf{x}\otimes\mathbf{I}_m) + \mathbf{K}_3]_2) \cdot e([-\mathbf{c}_2^\top]_1, [\mathbf{R}]_2) \cdot \boxed{e([\mathbf{c}_3^\top]_1, [\mathbf{r}\cdot\mathbf{e}_1^\top]_2)} \tag{18}$$
3. run $\mathbf{d}_{f,\mathbf{x}} \leftarrow \mathsf{rec}(f, \mathbf{x})$ (see Section 4.2), output
$$[D]_T = [(\mathbf{p}_1^\top, \mathbf{p}_2^\top)\mathbf{d}_{f,\mathbf{x}}]_T \tag{19}$$

Simulator. The simulator for Π_{ext} is as follows. The boxes indicate the changes from the simulator for Π_{one} in Section 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

$$\begin{array}{llll} \mathbf{A} \leftarrow \mathbb{Z}_p^{(k+1)\times k} & \text{and} \quad \mathbf{W} \leftarrow \mathbb{Z}_p^{(k+1)\times n'}, & \boxed{\mathbf{W}_0 \leftarrow \mathbb{Z}_p^{(k+1)\times k}}, & \mathbf{U} \leftarrow \mathbb{Z}_p^{(k+1)\times kn}, \ \mathbf{V} \leftarrow \mathbb{Z}_p^{(k+1)\times k} \\ \mathbf{c} \leftarrow \mathbb{Z}_p^{k+1} & \widetilde{\mathbf{w}} \leftarrow \mathbb{Z}_p^{n'}, & \boxed{\widetilde{\mathbf{w}}_0 \leftarrow \mathbb{Z}_p^{k}}, & \widetilde{\mathbf{v}} \leftarrow \mathbb{Z}_p^{k} \end{array}$$

and output

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

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

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

$$\mathsf{sk}^*_{f,\mathbf{r}} = \left(\mathbf{C}^\perp \cdot \mathsf{sk}^*_{f,\mathbf{r}}[1] + \mathbf{a}^\perp \cdot \mathsf{sk}^*_{f,\mathbf{r}}[2], [\mathbf{R}]_2\right) \quad \text{and} \quad (f, [\mathbf{r}]_2) \tag{20}$$

where

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

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

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

6.2 Analysis

The proof of correctness and security are the same as before in (12) and Lemma 4,6, with the following modifications (indicated by the boxes):

– In the proof of correctness, we have

$$\begin{aligned}[\mathbf{p}_2^\top]_T &= e([\mathbf{c}_0^\top]_1, [\mathbf{K}_2(\mathbf{x}\otimes\mathbf{I}_m)+\mathbf{K}_3]_2)\cdot e([-\mathbf{c}_2^\top]_1,[\mathbf{R}]_2)\cdot \boxed{e([\mathbf{c}_3^\top]_1,[\mathbf{r}\cdot\mathbf{e}_1^\top]_2)}\\ &= [\mathbf{s}^\top\mathbf{A}^\top\mathbf{T}\mathbf{L}_1(\mathbf{x}\otimes\mathbf{I}_m) - \boxed{\mathbf{s}^\top\mathbf{A}^\top\mathbf{W}_0\mathbf{r}\cdot\mathbf{e}_1^\top} + \mathbf{s}^\top\mathbf{A}^\top\mathbf{T}\mathbf{L}_0]_T \cdot \boxed{[\mathbf{w}^\top\mathbf{r}\cdot\mathbf{e}_1^\top + \mathbf{s}^\top\mathbf{A}^\top\mathbf{W}_0\mathbf{r}\cdot\mathbf{e}_1^\top]_T}\\ &= [\mathbf{s}^\top\mathbf{A}^\top\mathbf{T}\mathbf{L}_1(\mathbf{x}\otimes\mathbf{I}_m) + \mathbf{s}^\top\mathbf{A}^\top\mathbf{T}\mathbf{L}_0 + \boxed{\mathbf{w}^\top\mathbf{r}\cdot\mathbf{e}_1^\top}]_T\end{aligned}$$

and we rely on reconstruction with shift $\mathbf{w}^\top\mathbf{r}$.

– In Lemma 5, we also use

$$\overbrace{(\mathbf{A}^\top\mathbf{W}_0, \mathbf{c}^\top\mathbf{W}_0)}^{\mathsf{Game}_2} \equiv \overbrace{(\mathbf{A}^\top\mathbf{W}_0, \tilde{\mathbf{w}}_0^\top - (\mathbf{w}^*)^\top)}^{\mathsf{Game}_3}.$$

– Game_3, we have:

$$\mathsf{sk}_{f_j,\mathbf{r}_j}[2] = \big([\tilde{\mathbf{t}}_j^\top - (\mathbf{z}^*)^\top + \tilde{\mathbf{w}}^\top]_2, [\tilde{\mathbf{t}}_j^\top\mathbf{L}_{1,j} + \tilde{\mathbf{u}}^\top(\mathbf{I}_n\otimes\mathbf{R}_j)]_2, [\tilde{\mathbf{t}}_j^\top\mathbf{L}_{0,j} - \tilde{\mathbf{u}}^\top(\mathbf{I}_n\otimes\mathbf{R}_j)(\mathbf{x}^*\otimes\mathbf{I}_m) - \boxed{\tilde{\mathbf{w}}_0^\top\mathbf{r}_j\cdot\mathbf{e}_1^\top} + \tilde{\mathbf{v}}^\top\mathbf{R}_j]_2\big)$$

– Game_4, we have:

$$\mathsf{sk}_{f_j,\mathbf{r}_j}[2] = \big([\tilde{\mathbf{t}}_j^\top - (\mathbf{z}^*)^\top + \tilde{\mathbf{w}}^\top]_2, [\tilde{\mathbf{t}}_j^\top\mathbf{L}_{1,j} + \hat{\mathbf{u}}_j^\top]_2, [\tilde{\mathbf{t}}_j^\top\mathbf{L}_{0,j} - \hat{\mathbf{u}}_j^\top(\mathbf{x}^*\otimes\mathbf{I}_m) - \boxed{\tilde{\mathbf{w}}_0^\top\mathbf{r}_j\cdot\mathbf{e}_1^\top} + \tilde{\mathbf{v}}^\top\mathbf{R}_j]_2\big)$$

– In Lemma 7, recall that the difference between the two games lies in $\mathsf{sk}_{f_j,\mathbf{r}_j}[2]$:

$$\mathsf{sk}_{f_j,\mathbf{r}_j}[2] = \big([-\mathbf{p}_{j,1}^\top + \tilde{\mathbf{w}}^\top]_2, [\hat{\mathbf{u}}_j^\top]_2, [\mathbf{p}_{j,2}^\top - \hat{\mathbf{u}}_j^\top(\mathbf{x}^*\otimes\mathbf{I}_m) - \boxed{\tilde{\mathbf{w}}_0^\top\mathbf{r}_j\cdot\mathbf{e}_1^\top} + \tilde{\mathbf{v}}^\top\mathbf{R}]_2\big)$$

where we use

$$(\mathbf{p}_{j,1}^\top, \mathbf{p}_{j,2}^\top) \leftarrow \begin{cases} \mathsf{pgb}(f_j, \mathbf{x}^*, \mathbf{z}^*; \tilde{\mathbf{t}}_j) + \boxed{(\mathbf{0}, (\mathbf{w}^*)^\top\mathbf{r}_j\cdot\mathbf{e}_1^\top)} & \text{in } \mathsf{Game}_4 \\ \mathsf{pgb}^*(f_j, \mathbf{x}^*, f_j(\mathbf{x}^*)^\top\mathbf{z}^* + \boxed{(\mathbf{w}^*)^\top\mathbf{r}_j}; \tilde{\mathbf{t}}_j) & \text{in } \mathsf{Game}_5 \end{cases}$$

7 Π_{ubd}: Unbounded-Slot Scheme

In this section, we describe our unbounded-slot FE scheme. We give a generic transformation from scheme Π_{ext} in Section 6 and present a self-contained description of the scheme in Section B.

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 Section 6. Our unbounded-slot FE scheme Π_{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

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

We proceed as follows:

1. Compute
$$[D_1]_T \leftarrow \mathsf{Dec}_{\mathsf{ext}}\big((\mathsf{sk}_{f,1}, (f, [\mathbf{r}]_2)), (\mathsf{ct}_1, \mathbf{x}_1)\big); \tag{21}$$
2. For all $i \in [2, N]$, compute
$$[D_i]_T \leftarrow \mathsf{Dec}_{\mathsf{ext}}\big((\mathsf{sk}_{f,2}, (f, [\mathbf{r}]_2)), (\mathsf{ct}_i, \mathbf{x}_i)\big); \tag{22}$$
3. Compute
$$[D]_T = [D_1]_T \cdots [D_N]_T \tag{23}$$
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 sk_f with randomness $\mathbf{r}$, we have

$$D_1 = f(\mathbf{x}_1)^\top \mathbf{z}_1 - \textstyle\sum_{i\in[2,N]} \mathbf{w}_i^\top \mathbf{r} \tag{24}$$
$$D_i = f(\mathbf{x}_i)^\top \mathbf{z}_i + \mathbf{w}_i^\top \mathbf{r}, \qquad \forall i \in [2, N] \tag{25}$$
$$D = \textstyle\sum_{i\in[N]} f(\mathbf{x}_i)^\top \mathbf{z}_i \tag{26}$$

Here (24) and (25) follow from the correctness of Π_{ext} and the last (26) is implied by (24) and (25). 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 Π_{ext} achieves simulation-based semi-adaptive security, our unbounded-slot FE scheme Π_{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 Π_{ext}, we start by describing the simulator for Π_{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 Enc^* pass auxiliary information to 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

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

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

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

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

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

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*

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

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 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 Π_{ext}. See Lemma 8 for more details.

$\underline{\mathsf{Game}_{\eta.0}}$ for $\eta \in [2,N]$: Identical to 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

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

– 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

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

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

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 Π_{ext}. See Lemma 9 for more details.

$\underline{\mathsf{Game}_{\eta.2}}$ 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 $\mathrm{MDDH}_{k,Q}^1$ assumption: for all $f_j, \mathbf{x}_\eta^*, \mathbf{z}_\eta^*$,

$$\begin{aligned} &\Big\{ \overbrace{[-\mathbf{w}_\eta^\top \mathbf{r}_j]_2}^{\mathsf{sk}_{f_j,1}^*}, \overbrace{[\boxed{f_j(\mathbf{x}_\eta^*)^\top \mathbf{z}_\eta^*} + \mathbf{w}_\eta^\top \mathbf{r}_j]_2}^{\mathsf{sk}_{f_j,2}^*}, [\mathbf{r}_j]_2 \Big\}_{j\in[Q]} \\ \approx_c &\Big\{ [\boxed{f_j(\mathbf{x}_\eta^*)^\top \mathbf{z}_\eta^*} - \mathbf{w}_\eta^\top \mathbf{r}_j]_2, [\mathbf{w}_\eta^\top \mathbf{r}_j]_2, [\mathbf{r}_j]_2 \Big\}_{j\in[Q]} \end{aligned} \tag{27}$$

where $\mathbf{w}_\eta, \mathbf{r}_j \leftarrow \mathbb{Z}_p^k$ for all $j \in [Q]$. See Lemma 10 for more details.

$\underline{\mathsf{Game}_{\eta.3}}$ 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

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

– 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

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

– 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 Π_{ext} with the fact $f_j(\mathbf{x}_\eta^*)^\top \mathbf{0} + \mathbf{w}_\eta^\top \mathbf{r} = \mathbf{w}_\eta^\top \mathbf{r}$. See Lemma 11 for more details.

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.

7.4 Lemmas

We prove the following lemmas showing the indistinguishability of adjacent games listed above. This completes the proof of Theorem 4.

Lemma 8 ($\mathsf{Game}_0 \approx_c \mathsf{Game}_1$)**.** *For all $\mathcal{A}$, there exists $\mathcal{B}_1$ with $\mathsf{Time}(\mathcal{B}_1) \approx \mathsf{Time}(\mathcal{A})$ such that*

$$|\mathsf{Adv}_{\mathcal{A}}^1(\lambda) - \mathsf{Adv}_{\mathcal{A}}^0(\lambda)| \le \mathsf{Adv}_{\mathcal{B}_1}^{\Pi_{\mathsf{ext}}}(\lambda).$$

Proof. Recall that the difference lies in msk_1, ct_1 and $\mathsf{sk}_{f_j,1}$:

– in Game_0, we compute them by $\boxed{(\mathsf{Setup}_{\mathsf{ext}}, \mathsf{Enc}_{\mathsf{ext}}, \mathsf{KeyGen}_{\mathsf{ext}})}$;
– in Game_1, we compute them by $(\mathsf{Setup}_{\mathsf{ext}}^*, \mathsf{Enc}_{\mathsf{ext}}^*, \mathsf{KeyGen}_{\mathsf{ext}}^*)$.

The lemma follows from the simulation-based semi-adaptive security of Π_{ext}, cf. Section 4. The algorithm $\mathcal{B}_1$ works as follows:

Initialize. Get $\widehat{\mathsf{mpk}}$ from the challenger, run

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

and return $(\widehat{\mathsf{mpk}}, \mathsf{mpk}_2)$ to $\mathcal{A}$.

Challenge. On the challenge message $(\mathbf{x}_1^*, \mathbf{z}_1^*, \ldots, \mathbf{x}_N^*, \mathbf{z}_N^*)$ from $\mathcal{A}$, pick $\mathbf{w}_2, \ldots, \mathbf{w}_N \leftarrow \mathbb{Z}_p^k$, submit $(\mathbf{x}_1^*, \mathbf{z}_1^* \| -\sum_{i\in[2,N]} \mathbf{w}_i)$ to the challenger and get $\widehat{\mathsf{ct}}$. Run

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

and return $\mathsf{ct}^* = (\widehat{\mathsf{ct}}, \mathsf{ct}_2, \ldots, \mathsf{ct}_N)$ to $\mathcal{A}$.

Key queries. On input the j-th key query f_j, sample $\mathbf{r}_j \leftarrow \mathbb{Z}_p^k$, and compute

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

Send a key query $(f_j, [\mathbf{r}_j]_2)$ to the challenger and get back $\widehat{\mathsf{sk}}_j$. Output $\mathsf{sk}_{f_j} = (\widehat{\mathsf{sk}}_j, \mathsf{sk}_{f_j,2}, [\mathbf{r}_j]_2)$.

Game	ct^*				sk_f		**Remark**
	ct_1	$\mathsf{ct}_i, 1 < i < \eta$	ct_η	$\mathsf{ct}_i, \eta < i \le N$	$\mathsf{sk}_{f,1}$	$\mathsf{sk}_{f,2}$	
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:	Π_{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: $[f(\mathbf{x}_\eta^*)^\top \mathbf{z}_\eta^* + \mathbf{w}_\eta^\top \mathbf{r}]_2$	Π_{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: $[\sum_{i\le\eta} f(\mathbf{x}_i^*)^\top \mathbf{z}_i^* - \sum \mathbf{w}_i^\top \mathbf{r}]_2$	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: $[\sum_{i\le\eta} f(\mathbf{x}_i^*)^\top \mathbf{z}_i^* - \sum \mathbf{w}_i^\top \mathbf{r}]_2$	real:	Π_{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 Π_{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 $\in$ {real, 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 $i \in [2, N]$.

Analysis. Observe that

- when $(\widehat{\mathsf{mpk}}, \widehat{\mathsf{msk}}) = (\mathsf{mpk}_1, \boxed{\mathsf{msk}_1}) \leftarrow \boxed{\mathsf{Setup}_{\mathsf{ext}}}(1^\lambda, 1^n, 1^{n'})$ and

$$\widehat{\mathsf{ct}} \leftarrow \boxed{\mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_1, (\mathbf{x}_1^*, \mathbf{z}_1^* \| -\textstyle\sum_{i\in[2,N]} \mathbf{w}_i))}$$

$$\widehat{\mathsf{sk}}_j \leftarrow \boxed{\mathsf{KeyGen}_{\mathsf{ext}}(\mathsf{msk}_1, (f_j, [\mathbf{r}_j]_2))}, \forall\, j \in [Q]$$

  the simulation is identical to Game_0;
- when $(\widehat{\mathsf{mpk}}, \widehat{\mathsf{msk}}) = (\mathsf{mpk}_1, \mathsf{msk}_1^*) \leftarrow \mathsf{Setup}_{\mathsf{ext}}^*(1^\lambda, 1^n, 1^{n'})$ and

$$\widehat{\mathsf{ct}} \leftarrow \mathsf{Enc}_{\mathsf{ext}}^*(\mathsf{msk}_1^*, \mathbf{x}_1^*)$$

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

  the simulation is identical to Game_1.

This readily proves the lemma. □

Lemma 9 ($\mathsf{Game}_{\eta.0} \approx_c \mathsf{Game}_{\eta.1}$)**.** *For all $\mathcal{A}$ and all $\eta \in [2, N]$, there exists $\mathcal{B}_1$ with $\mathsf{Time}(\mathcal{B}_1) \approx \mathsf{Time}(\mathcal{A})$ such that*

$$|\mathsf{Adv}_{\mathcal{A}}^{\eta.1}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{\eta.0}(\lambda)| \le \mathsf{Adv}_{\mathcal{B}_1}^{\Pi_{\mathsf{ext}}}(\lambda).$$

Proof. Recall that the difference lies in msk_2, ct_η and $\mathsf{sk}_{f_j,1}$:

- in $\mathsf{Game}_{\eta.0}$, we compute them by $\boxed{(\mathsf{Setup}_{\mathsf{ext}}, \mathsf{Enc}_{\mathsf{ext}}, \mathsf{KeyGen}_{\mathsf{ext}})}$;
- in $\mathsf{Game}_{\eta.1}$, we compute them by $(\mathsf{Setup}_{\mathsf{ext}}^*, \mathsf{Enc}_{\mathsf{ext}}^*, \mathsf{KeyGen}_{\mathsf{ext}}^*)$.

The lemma follows from the simulation-based semi-adaptive security of Π_{ext}, cf. Section 4. The algorithm $\mathcal{B}_1$ works as follows:

Initialize. Get $\widehat{\mathsf{mpk}}$ from the challenger, run

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

and return $(\mathsf{mpk}_1, \widehat{\mathsf{mpk}})$ to $\mathcal{A}$.

Challenge. On input the challenge message $(\mathbf{x}_1^*, \mathbf{z}_1^*, \ldots, \mathbf{x}_N^*, \mathbf{z}_N^*)$ from $\mathcal{A}$, pick $\mathbf{w}_2, \ldots, \mathbf{w}_N \leftarrow \mathbb{Z}_p^k$, submit $(\mathbf{x}_\eta^*, \mathbf{z}_\eta^* \| \mathbf{w}_\eta)$ to the challenger and get back $\widehat{\mathsf{ct}}$. Compute

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

and output $\mathsf{ct}^* = (\mathsf{ct}_1^*, \mathsf{ct}_2, \ldots, \mathsf{ct}_N)$.

Key queries. On input the j-th key queries f_j, sample $\mathbf{r}_j \leftarrow \mathbb{Z}_p^k$ and compute

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

Send a key query $(f_j, [\mathbf{r}_j]_2)$ to the challenger and get back $\widehat{\mathsf{sk}}_j$. Output $\mathsf{sk}_{f_j} = (\mathsf{sk}_{f_j,1}^*, \widehat{\mathsf{sk}}_j, [\mathbf{r}_j]_2)$.

Analysis. Observe that,

- when $(\widehat{\mathsf{mpk}}, \widehat{\mathsf{msk}}) = (\mathsf{mpk}_2, \boxed{\mathsf{msk}_2}) \leftarrow \boxed{\mathsf{Setup}_{\mathsf{ext}}}(1^\lambda, 1^n, 1^{n'})$ and

$$\widehat{\mathsf{ct}} \leftarrow \boxed{\mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}^*_\eta, \mathbf{z}^*_\eta \| \mathbf{w}_\eta))}, \quad \widehat{\mathsf{sk}}_j \leftarrow \boxed{\mathsf{KeyGen}_{\mathsf{ext}}(\mathsf{msk}_2, (f_j, [\mathbf{r}_j]_2))}, \forall\, j \in [Q]$$

  the simulation is identical to $\mathsf{Game}_{\eta.0}$;
- when $(\widehat{\mathsf{mpk}}, \widehat{\mathsf{msk}}) = (\mathsf{mpk}_2, \mathsf{msk}^*_2) \leftarrow \mathsf{Setup}^*_{\mathsf{ext}}(1^\lambda, 1^n, 1^{n'})$ and

$$\widehat{\mathsf{ct}} \leftarrow \mathsf{Enc}^*_{\mathsf{ext}}(\mathsf{msk}^*_2, \mathbf{x}^*_\eta), \quad \widehat{\mathsf{sk}}_j \leftarrow \mathsf{KeyGen}^*_{\mathsf{ext}}(\mathsf{msk}^*_2, \mathbf{x}^*_\eta, (f_j, [\mathbf{r}_j]_2), [f_j(\mathbf{x}^*_\eta)^\top \mathbf{z}^*_\eta + \mathbf{w}_\eta^\top \mathbf{r}_j]_2), \forall\, j \in [Q]$$

  the simulation is identical to $\mathsf{Game}_{\eta.1}$.

This readily proves the lemma. □

Lemma 10 ($\mathsf{Game}_{\eta.1} \approx_c \mathsf{Game}_{\eta.2}$)**.** *For all $\mathcal{A}$ and all $\eta \in [2, N]$, there exists $\mathcal{B}_2$ with $\mathsf{Time}(\mathcal{B}_2) \approx \mathsf{Time}(\mathcal{A})$ such that*

$$|\mathsf{Adv}^{\eta.2}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{\eta.1}_{\mathcal{A}}(\lambda)| \le 2 \cdot \mathsf{Adv}^{\mathrm{MDDH}^1_{k,Q}}_{\mathcal{B}_2}(\lambda)$$

where Q is the number of key queries.

Proof. Recall that the difference lies in $\mathsf{sk}^*_{f_j,1}$ and $\mathsf{sk}^*_{f_j,2}$. We prove the lemma using Lemma 1 w.r.t. $\mathbf{w}_\eta$ and $f_j(\mathbf{x}^*_\eta)^\top \mathbf{z}^*_\eta$. On input $([t_{j,1}]_2, [t_{j,2}]_2, [\mathbf{r}_j]_2)$, the reduction $\mathcal{B}_2$ sample $\mathbf{w}_i \leftarrow \mathbb{Z}_p^k$ for $i \in [2, N] \setminus \{\eta\}$ and run

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

this allows us to simulate mpk and create challenge ciphertext. Here we crucially use the fact that we do not need $\mathbf{w}_\eta$ when generating the ciphertext. For the j-th key queries f_j, we compute

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

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

using the term in the input and output $\mathsf{sk}_{f_j} = (\mathsf{sk}^*_{f_j,1}, \mathsf{sk}^*_{f_j,2}, [\mathbf{r}_j]_2)$. This proves the lemma. □

Lemma 11 ($\mathsf{Game}_{\eta.2} \approx_c \mathsf{Game}_{\eta.3}$)**.** *For all $\mathcal{A}$ and all $\eta \in [2, N]$, there exists $\mathcal{B}_1$ with $\mathsf{Time}(\mathcal{B}_1) \approx \mathsf{Time}(\mathcal{A})$ such that*

$$|\mathsf{Adv}^{\eta.3}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{\eta.2}_{\mathcal{A}}(\lambda)| \le \mathsf{Adv}^{\Pi_{\mathsf{ext}}}_{\mathcal{B}_1}(\lambda).$$

Proof. The proof is analogous to that for Lemma 9 with the following modifications: $\mathcal{B}_1$ submits $(\mathbf{x}^*_\eta, \mathbf{0} \| \mathbf{w}_\eta)$ to the challenger during **Challenge**, so that

- when $(\widehat{\mathsf{mpk}}, \widehat{\mathsf{msk}}) = (\mathsf{mpk}_2, \boxed{\mathsf{msk}_2}) \leftarrow \boxed{\mathsf{Setup}_{\mathsf{ext}}}(1^\lambda, 1^n, 1^{n'})$ and

$$\widehat{\mathsf{ct}} \leftarrow \boxed{\mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}^*_\eta, \mathbf{0} \| \mathbf{w}_\eta))}, \quad \widehat{\mathsf{sk}}_j \leftarrow \boxed{\mathsf{KeyGen}_{\mathsf{ext}}(\mathsf{msk}_2, (f_j, [\mathbf{r}_j]_2))}, \forall\, j \in [Q]$$

  the simulation is identical to $\mathsf{Game}_{\eta.3}$;
- when $(\widehat{\mathsf{mpk}}, \widehat{\mathsf{msk}}) = (\mathsf{mpk}_2, \mathsf{msk}^*_2) \leftarrow \mathsf{Setup}^*_{\mathsf{ext}}(1^\lambda, 1^n, 1^{n'})$ and

$$\widehat{\mathsf{ct}} \leftarrow \mathsf{Enc}^*_{\mathsf{ext}}(\mathsf{msk}^*_2, \mathbf{x}^*_\eta), \quad \widehat{\mathsf{sk}}_j \leftarrow \mathsf{KeyGen}^*_{\mathsf{ext}}(\mathsf{msk}^*_2, \mathbf{x}^*_\eta, (f_j, [\mathbf{r}_j]_2), [\mathbf{w}_\eta^\top \mathbf{r}_j]_2), \forall\, j \in [Q]$$

  the simulation is identical to $\mathsf{Game}_{\eta.2}$. □

8 Π_{mcl}: Multi-Client Scheme

In this section, we introduce the notion of multi-client unbounded-slot FE for attribute-weighted sums and provide a generic scheme Π_{mcl} based on Π_{ext} as in Π_{ubd}. We refer the reader to Section 1.2 for a quick introduction.

Overview. For simplicity, we consider one client per slot, though the definition and the construction extend readily to the setting where there are fewer clients than slots, where the complexity of the one-time secret key generation grows only with the number of clients and not the number of slots.

Comparison with prior works. We proceed to comparisons with the notion of multi-client FE in prior works [23,1,2]:

- Each client in our scheme does not have any *long-term* secret key for encryption from the authority; they merely see a mpk from the authority.
- Instead, a group of clients generate a set of *one-time* secret keys by themselves for *each* encryption; the distribution is independent of the databases and does not require interaction with the authority. We consider only static corruption of clients that leak the one-time key.
- The one-time secret keys also rule out mix-and-match attacks amongst the ciphertexts, which limit the power of the adversary.

Overview of the construction. Roughly speaking, we would like to show that the unbounded-slot FE scheme Π_{ubd} in the previous section satisfies the following property: for all $S \subsetneq [N]$, we can simulate

$$\mathsf{mpk}, \mathsf{ct}, \mathsf{sk}_f, (\mathbf{w}_i)_{i\in S}$$

given $\sum_{i\notin S} f(\mathbf{x}_i)^\top \mathbf{z}_i$. That is, the leakage on $(\mathbf{z}_i)_{i\notin S}$ is bounded by the attribute-weighted sums computed over these entries in the database. Turns out this is indeed true if $1 \notin S$ so that $\mathbf{w}_1$ remains hidden, via a straight-forward adaptation of the previous proof. To avoid this restriction on S, we need to modify the scheme to break the asymmetry of the construction w.r.t. the first slot.

8.1 Definition

A multi-client unbounded-slot FE for attribute-weighted sums is the same as an unbounded-slot FE for the same function class with the following changes:

- $\mathsf{OTSKGen}$ takes input $1^\lambda, 1^N$ and outputs $(\mathsf{otsk}_i)_{i\in[N]}$.
- Enc takes input $\mathsf{mpk}, (\mathbf{x}, \mathbf{z})$ and an additional input otsk.
- Dec takes input (sk_f, f) and $(\mathsf{ct}_i, \mathbf{x}_i)_{i\in[N]}$.

Both correctness and security can be adapted from those for unbounded-slot FE, see below.

Correctness. For all N, $(\mathbf{x}_i, \mathbf{z}_i)_{i\in[N]}$ and for all $f \in \mathcal{F}_{\mathsf{ABP},n,n'}$, we require

$$\Pr\left[\mathsf{Dec}((\mathsf{sk}_f, f), (\mathsf{ct}_i, \mathbf{x}_i)_{i\in[N]}) = \sum_{i=1}^N f(\mathbf{x}_i)^\top \mathbf{z}_i : \begin{array}{l}(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^{n'}) \\ (\mathsf{otsk}_i)_{i\in[N]} \leftarrow \mathsf{OTSKGen}(1^\lambda, 1^N) \\ \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f) \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{otsk}_i, (\mathbf{x}_i, \mathbf{z}_i)), i \in [N]\end{array}\right] = 1.$$

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

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

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

8.2 Construction

Scheme. Assume $\Pi_{\mathsf{ext}} = (\mathsf{Setup}_{\mathsf{ext}}, \mathsf{Enc}_{\mathsf{ext}}, \mathsf{KeyGen}_{\mathsf{ext}}, \mathsf{Dec}_{\mathsf{ext}})$ be the extended one-slot scheme in Section 6. Our multi-client unbounded-slot FE Π_{mcl} is similar to Π_{ubd} in Section 7:

- $\mathsf{Setup}, \mathsf{KeyGen}$ as in Π_{ubd}.
- $\mathsf{OTSKGen}(1^\lambda, 1^N)$: Sample $\mathbf{w}_i \leftarrow \mathbb{Z}_p^k$ for all $i \in [N]$ subject to the constraint $\sum_{i\in[N]} \mathbf{w}_i = \mathbf{0}$ and output

$$(\mathsf{otsk}_i = \mathbf{w}_i)_{i\in[N]}.$$

- $\mathsf{Enc}(\mathsf{mpk}, \mathsf{otsk}, (\mathbf{x}, \mathbf{z}))$: Parse $\mathsf{otsk} = \mathbf{w}$. Sample $\mathbf{z}' \leftarrow \mathbb{Z}_p^{n'}$, $\mathbf{w}' \leftarrow \mathbb{Z}_p^k$ and compute

$$\begin{aligned}\mathsf{ct}_1 &\leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_1, (\mathbf{x}, \mathbf{z}' \| \mathbf{w}'))\\ \mathsf{ct}_2 &\leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}, \mathbf{z} - \mathbf{z}' \| \mathbf{w} - \mathbf{w}'))\end{aligned}$$

  and output

$$\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2) \quad \text{and} \quad \mathbf{x}.$$

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

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

  We proceed as follows:

  1. For all $i \in [N]$, compute

$$\begin{aligned}[D_{i,1}]_T &\leftarrow \mathsf{Dec}_{\mathsf{ext}}\big((\mathsf{sk}_{f,1}, (f, [\mathbf{r}]_2)), (\mathsf{ct}_{i,1}, \mathbf{x}_i)\big);\\ [D_{i,2}]_T &\leftarrow \mathsf{Dec}_{\mathsf{ext}}\big((\mathsf{sk}_{f,2}, (f, [\mathbf{r}]_2)), (\mathsf{ct}_{i,2}, \mathbf{x}_i)\big);\end{aligned}$$

  2. Compute

$$[D]_T = \prod_{i\in[N]} ([D_{i,1}]_T \cdot [D_{i,2}]_T)$$

     and output D via brute-force discrete log.

Correctness. Correctness uses

$$\sum_{i\in[N]} f(\mathbf{x}_i)^\top \mathbf{z}_i = \sum_{i\in[N]} \big(\overbrace{f(\mathbf{x}_i)^\top \mathbf{z}'_i + (\mathbf{w}'_i)^\top}^{D_{i,1}} \mathbf{r} + \overbrace{f(\mathbf{x}_i)^\top (\mathbf{z}_i - \mathbf{z}'_i) + (\mathbf{w}_i - \mathbf{w}'_i)}^{D_{i,2}} \mathbf{r} \big)$$

which in turn uses $\sum \mathbf{w}_i = \mathbf{0}$.

Security. We prove the following theorem.

Theorem 5. *Assume that extended one-slot scheme Π_{ext} achieves simulation-based semi-adaptive security, our multi-client unbounded-slot FE scheme Π_{mcl} described in this section achieves simulation-based semi-adaptive security under the k-Linear assumption in $\mathbb{G}_2$.*

Both simulator and proof are analogous to those of Π_{ubd} in Section 7 with a high-level idea as follows. WLOG, assume $\bar{S} = [N']$ for some $N' < N$. We can rewrite $\mathsf{ct}_1 = (\mathsf{ct}_{1,1}, \mathsf{ct}_{1,2})$ as

$$\mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_1, (\mathbf{x}_1^*, \mathbf{z}_1^* - \mathbf{z}'_1 \| \mathbf{w}_1 - \mathbf{w}'_1)),\ \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}_1^*, \mathbf{z}'_1 \| \mathbf{w}'_1))$$

by relying on the entropy in $\mathbf{z}'_1, \mathbf{w}'_1$. For notational simplicity, for the rest of the overview, we assume $\mathbf{w}'_i = \mathbf{0}$ and $\mathbf{z}'_i = \mathbf{0}$ for all $i \in \bar{S}$; the general case follows readily. That is, we only rely on the entropy in the $(\mathbf{z}'_i, \mathbf{w}'_i)$'s to rewrite $\mathsf{ct}_1 = (\mathsf{ct}_{1,1}, \mathsf{ct}_{1,2})$. We now have:

$$\mathsf{ct}^*_i = (\mathsf{ct}^*_{i,1}, \mathsf{ct}^*_{i,2}) = \begin{cases} \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_1, (\mathbf{x}^*_1, \mathbf{z}^*_1 \| \mathbf{w}_1)), \ \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}^*_1, \mathbf{0} \| \mathbf{0})) & i = 1 \\ \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_1, (\mathbf{x}^*_i, \mathbf{0} \| \mathbf{0})), \quad \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}^*_i, \mathbf{z}^*_i \| \mathbf{w}_i)) & 1 < i \le N' \end{cases}$$

We can then basically follow the idea for Π_{ubd} in Section 7 by the following observation:

$$\mathsf{ct}^*_{1,1} = \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_1, (\mathbf{x}^*_1, \mathbf{z}^*_1 \| \mathbf{w}_1)), \quad \{\mathsf{ct}^*_{i,2} = \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}^*_i, \mathbf{z}^*_i \| \mathbf{w}_i))\}_{1<i\le N'}$$

looks like a ciphertext in Π_{ubd} and KeyGen is exactly the same as in Π_{ubd}. We note that:

- in the simulator and throughout the proof, we don't change

  $$\mathsf{ct}^*_{1,2} = \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}^*_1, \mathbf{0} \| \mathbf{0})), \quad \{\mathsf{ct}^*_{i,1} = \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_1, (\mathbf{x}^*_i, \mathbf{0} \| \mathbf{0}))\}_{1<i\le N};$$

  and the reduction can always simulate them using $\mathsf{mpk}_1, \mathsf{mpk}_2$;
- As before, the proof also use the fact that for all $i \in [2, N']$ and $\mu \in \mathbb{Z}_p$, it holds that

  $$\{[-\mathbf{w}_i^\top \mathbf{r}_j]_2, [\mu + \mathbf{w}_i^\top \mathbf{r}_j]_2, [\mathbf{r}_j]_2\}_{j\in[Q]} \approx_c \{[\mu - \mathbf{w}_i^\top \mathbf{r}_j]_2, [\mathbf{w}_i^\top \mathbf{r}_j]_2, [\mathbf{r}_j]_2\}_{j\in[Q]}$$

  here we crucially relies on the fact that $i \notin S$ and $\mathbf{w}_i$ is not leaked to the adversary; otherwise the above statement does not hold.

In the actual proof with general S, instead of enumerating over $i \in [2, N']$ in the games, we enumerate $i \in \bar{S} \setminus \{i^*\}$ where $i^* = \min\{\bar{S}\}$.

8.3 Simulator

Let S be the set indicating corrupted clients and $i^* := \min\{\bar{S}\}$, that is, the smallest value outside S. (Simulation is trivial if $S = [N]$.) We describe the simulator for Π_{mcl} using the simulator for Π_{ext}, i.e., $(\mathsf{Setup}^*_{\mathsf{ext}}, \mathsf{Enc}^*_{\mathsf{ext}}, \mathsf{KeyGen}^*_{\mathsf{ext}})$, as in Section 7.2; also, the adversary needs to commit to the length N in advance.

- $\mathsf{Setup}^*(1^\lambda, 1^n, 1^{n'}, 1^N, S)$: Sample

  $$\mathbf{z}'_1, \ldots, \mathbf{z}'_N \leftarrow \mathbb{Z}_p^{n'}, \mathbf{w}'_1, \ldots, \mathbf{w}'_N \leftarrow \mathbb{Z}_p^k \quad \text{and} \quad \mathbf{w}_1, \ldots, \mathbf{w}_N \leftarrow \mathbb{Z}_p^k \text{ such that } \sum_{i\in[N]} \mathbf{w}_i = \mathbf{0};$$

  run

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

  and output

  $$\mathsf{mpk} = (\mathsf{mpk}_1, \mathsf{mpk}_2) \quad \text{and} \quad \mathsf{msk}^* = (\mathsf{msk}^*_1, \mathsf{msk}_2, (\mathbf{w}_i, \mathbf{w}'_i, \mathbf{z}'_i)_{i\in[N]}, S).$$

- $\mathsf{OTSKGen}^*(\mathsf{msk}^*)$: Parse $\mathsf{msk}^* = (\mathsf{msk}^*_1, \mathsf{msk}_2, (\mathbf{w}_i, \mathbf{w}'_i, \mathbf{z}'_i)_{i\in[N]}, S)$ and output $(\mathsf{otsk}_i = \mathbf{w}_i)_{i\in S}$.
- $\mathsf{Enc}^*(\mathsf{msk}^*, (\mathbf{x}^*_i)_{i\notin S})$: Compute

  $$\begin{cases} \mathsf{ct}^*_{i^*,1} \leftarrow \mathsf{Enc}^*_{\mathsf{ext}}(\mathsf{msk}^*_1, \mathbf{x}^*_{i^*}), & \mathsf{ct}^*_{i^*,2} \leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}^*_{i^*}, \mathbf{z}'_{i^*} \| \mathbf{w}'_{i^*})), \\ \mathsf{ct}^*_{i,1} \leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_1, (\mathbf{x}^*_i, \mathbf{z}'_i \| \mathbf{w}'_i)), & \mathsf{ct}^*_{i,2} \leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}^*_i, -\mathbf{z}'_i \| \mathbf{w}_i - \mathbf{w}'_i)), \ \forall\, i \in \bar{S} \setminus \{i^*\} \end{cases}$$

  and output

  $$(\mathsf{ct}^*_i = (\mathsf{ct}^*_{i,1}, \mathsf{ct}^*_{i,2}))_{i\in\bar{S}} \quad \text{and} \quad (\mathbf{x}^*_i)_{i\in\bar{S}}.$$

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

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

and output

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

Remark 4 (decryption). Observe that when we decrypt the simulated ciphertexts:

$$(\mathsf{ct}_{i,1}^*, \mathsf{ct}_{i,2}^*)_{i\in\bar{S}} \leftarrow \mathsf{Enc}^*(\mathsf{msk}^*, (\mathbf{x}_i^*)_{i\notin S}) \quad \text{and} \quad (\mathsf{ct}_{i,1}^*, \mathsf{ct}_{i,2}^*) \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{otsk}_i, (\mathbf{x}_i^*, \mathbf{z}_i^*)), \forall i \in S$$

with the simulated secret key $(\mathsf{sk}_{f,1}^*, \mathsf{sk}_{f,2}^*, [\mathbf{r}]_2) \leftarrow \mathsf{KeyGen}^*(\mathsf{msk}^*, (\mathbf{x}_i^*)_{i\notin S}, f, \sum_{i\notin S} f(\mathbf{x}_i^*)^\top \mathbf{z}_i^*)$, we get

$$\begin{aligned}
D_{i^*,1} &= \textstyle\sum_{i\notin S} f(\mathbf{x}_i)^\top \mathbf{z}_i^* - f(\mathbf{x}_{i^*}^*)^\top \mathbf{z}'_{i^*} + (\mathbf{w}_{i^*} - \mathbf{w}'_{i^*})^\top \mathbf{r}, & D_{i^*,2} &= f(\mathbf{x}_{i^*}^*)^\top \mathbf{z}'_{i^*} + (\mathbf{w}'_{i^*})^\top \mathbf{r}, \\
D_{i,1} &= f(\mathbf{x}_i^*)^\top \mathbf{z}'_i + (\mathbf{w}'_i)^\top \mathbf{r}, & D_{i,2} &= -f(\mathbf{x}_i^*)^\top \mathbf{z}'_i + (\mathbf{w}_i - \mathbf{w}'_i)^\top \mathbf{r}, \qquad \forall\, i \in \bar{S}\setminus\{i^*\} \\
D_{i,1} &= f(\mathbf{x}_i^*)^\top \mathbf{z}'_i + (\mathbf{w}'_i)^\top \mathbf{r}, & D_{i,2} &= f(\mathbf{x}_i^*)^\top (\mathbf{z}_i^* - \mathbf{z}'_i) + (\mathbf{w}_i - \mathbf{w}'_i)^\top \mathbf{r}, \ \forall\, i \in S
\end{aligned}$$

and end up with

$$D = \sum_{i\in[N]} (D_{i,1} + D_{i,2}) = \sum_{i\in[N]} f(\mathbf{x}_i^*)^\top \mathbf{z}_i^* + \sum_{i\in[N]} \mathbf{w}_i^\top \mathbf{r} = \sum_{i\in[N]} f(\mathbf{x}_i^*)^\top \mathbf{z}_i^*$$

Furthermore, with the corruption of $(\mathsf{otsk}_i = \mathbf{w}_i)_{i\in S}$, we can recover

$$\sum_{i\notin S} (D_{i,1} + D_{i,2}) + \sum_{i\in S} \mathbf{w}_i^\top \mathbf{r} = (\sum_{i\notin S} f(\mathbf{x}_i^*)^\top \mathbf{z}_i^* + \sum_{i\notin S} \mathbf{w}_i^\top \mathbf{r}) + \sum_{i\in S} \mathbf{w}_i^\top \mathbf{r} = \sum_{i\notin S} f(\mathbf{x}_i^*)^\top \mathbf{z}_i^*$$

Similarly, when we decrypt normal ciphertexts

$$(\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2}) \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{otsk}_i, (\mathbf{x}_i^*, \mathbf{z}_i^*)), \forall i \in [N]$$

with the simulated secret key $(\mathsf{sk}_{f,1}^*, \mathsf{sk}_{f,2}^*, [\mathbf{r}]_2)$, we get

$$D_{i,1} = f(\mathbf{x}_i^*)^\top \mathbf{z}'_i + (\mathbf{w}'_i)^\top \mathbf{r}, \quad D_{i,2} = f(\mathbf{x}_i^*)^\top (\mathbf{z}_i^* - \mathbf{z}'_i) + (\mathbf{w}_i - \mathbf{w}'_i)^\top \mathbf{r}, \forall\, i \in [N]$$

and end up with $\sum_{i\in[N]} f(\mathbf{x}_i^*)^\top \mathbf{z}_i^*$.

8.4 Proof sketch

Let $S \subsetneq [N]$ be the set of corrupted clients and $i^* := \min\{\bar{S}\}$, that is, the smallest value outside S. We use $(\mathbf{x}_i^*, \mathbf{z}_i^*)_{i\notin S}$ to denote the challenge. Our proof basically follows the proof for Π_{ubd} in Section 7 with an extra games and some modifications below. Also see a summary of game sequence in Fig 5.

An extra game: $\mathsf{Game}_{0'}$. After Game_0, we introduce an extra game where we generate $\mathsf{ct}_{i^*}^* = (\mathsf{ct}_{i^*,1}, \mathsf{ct}_{i^*,2})$ as follows:

$$\mathsf{ct}_{i^*,1} \leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_1, (\mathbf{x}_{i^*}^*, \boxed{\mathbf{z}_{i^*}^* - \mathbf{z}'_{i^*} \| \mathbf{w}_{i^*} - \mathbf{w}'_{i^*}})), \ \mathsf{ct}_{i^*,2} \leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}_{i^*}, \boxed{\mathbf{z}'_{i^*} \| \mathbf{w}'_{i^*}}))$$

instead of

$$\mathsf{ct}_{i^*,1} \leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_1, (\mathbf{x}_{i^*}^*, \boxed{\mathbf{z}'_{i^*} \| \mathbf{w}'_{i^*}})), \ \mathsf{ct}_{i^*,2} \leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}_{i^*}^*, \boxed{\mathbf{z}_{i^*}^* - \mathbf{z}'_{i^*} \| \mathbf{w}_{i^*} - \mathbf{w}'_{i^*}}))$$

We can show $\mathsf{Game}_0 \approx_s \mathsf{Game}_{0'}$ by the change of variables: $\mathbf{z}'_{i^*} \mapsto \mathbf{z}_{i^*}^* - \mathbf{z}'_{i^*}$ and $\mathbf{w}'_{i^*} \mapsto \mathbf{w}_{i^*} - \mathbf{w}'_{i^*}$.

Modifications. Now we have

$$\mathsf{ct}_i^* = \begin{cases} \mathsf{ct}_{i^*,1} \leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_1, (\mathbf{x}_{i^*}^*, \mathbf{z}_{i^*}^* - \mathbf{z}_{i^*}' \| \mathbf{w}_{i^*} - \mathbf{w}_{i^*}')), \ \mathsf{ct}_{i^*,2} \leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}_{i^*}^*, \mathbf{z}_{i^*}' \| \mathbf{w}_{i^*}')) & i = i^* \\ \mathsf{ct}_{i,1} \leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_1, (\mathbf{x}_i^*, \mathbf{z}_i' \| \mathbf{w}_i')), \quad \mathsf{ct}_{i,2} \leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}_i^*, \mathbf{z}_i^* - \mathbf{z}_i' \| \mathbf{w}_i - \mathbf{w}_i')) & i \in \bar{S} \setminus \{i^*\} \end{cases}$$

and then our game sequence mirrors that for Π_{ubd} in Section 7: we have

- Game_{i^*} in the place of Game_1; and
- $\mathsf{Game}_{\eta.0}, \mathsf{Game}_{\eta.1}, \mathsf{Game}_{\eta.2}, \mathsf{Game}_{\eta.3}$ enumerating over $\eta \notin \bar{S} \setminus \{i^*\}$ instead of $\eta \in [2, N]$.

Among all games, we make changes to master key pair $((\mathsf{mpk}_1, \mathsf{mpk}_2), (\mathsf{msk}_1, \mathsf{msk}_2))$, secret keys $\mathsf{sk}_{f_j} = (\mathsf{sk}_{f_j,1}, \mathsf{sk}_{f_j,2})$ and $(\mathsf{ct}_{i^*,1}, (\mathsf{ct}_{i,2})_{i \in \bar{S} \setminus \{i^*\}})$ which involves $(\mathbf{x}_i^*, \mathbf{z}_i^*)_{i \notin S}$; note that, we will never change $(\mathsf{ct}_{i^*,2}, (\mathsf{ct}_{i,1})_{i \in \bar{S} \setminus \{i^*\}})$. In more detail,

- in Game_{i^*}, we have $(\mathsf{mpk}_1, \mathsf{msk}_1^*) \leftarrow \mathsf{Setup}_{\mathsf{ext}}^*(1^\lambda, 1^n, 1^{n'})$ and compute

$$\mathsf{ct}_{i_1^*} \leftarrow \mathsf{Enc}_{\mathsf{ext}}^*(\mathsf{msk}_1^*, \mathbf{x}_{i^*}^*),$$

$$\mathsf{sk}_{f_j,1} \leftarrow \mathsf{KeyGen}_{\mathsf{ext}}^*\big(\mathsf{msk}_1^*, \mathbf{x}_{i^*}^*, (f_j, [\mathbf{r}_j]_2), [f_j(\mathbf{x}_{i^*}^*)^\top \mathbf{z}_{i^*}^* - f_j(\mathbf{x}_{i^*}^*)^\top \mathbf{z}_{i^*}' + \mathbf{w}_{i^*}^\top \mathbf{r} - (\mathbf{w}_{i^*}')^\top \mathbf{r}]_2\big)$$

  instead of having $(\mathsf{mpk}_1, \mathsf{msk}_1) \leftarrow \mathsf{Setup}_{\mathsf{ext}}^*(1^\lambda, 1^n, 1^{n'})$ and computing

$$\mathsf{ct}_{i_1^*} \leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_1, (\mathbf{x}_{i^*}^*, \mathbf{z}_{i^*}^* - \mathbf{z}_{i^*}' \| \mathbf{w}_{i^*} - \mathbf{w}_{i^*}')), \mathsf{sk}_{f_j,1} \leftarrow \mathsf{KeyGen}_{\mathsf{ext}}(\mathsf{msk}_1, (f_j, [\mathbf{r}]_2))$$

  The simulation-based security of Π_{ext} ensures that $\mathsf{Game}_{0'} \approx_c \mathsf{Game}_{i^*}$. The proof is analogous to that of Lemma 8.
- in $\mathsf{Game}_{\eta.0}$, we compute

$$\mathsf{ct}_{i,2} \leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}_i^*, \boxed{-\mathbf{z}_i'} \| \mathbf{w}_i - \mathbf{w}_i')), \forall i \notin S \wedge i < \eta$$

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

- in $\mathsf{Game}_{\eta.1}$, we have $(\mathsf{mpk}_2, \mathsf{msk}_2^*) \leftarrow \mathsf{Setup}_{\mathsf{ext}}^*(1^\lambda, 1^n, 1^{n'})$ and compute

$$\mathsf{ct}_{\eta,2} \leftarrow \mathsf{Enc}_{\mathsf{ext}}^*(\mathsf{msk}_2^*, \mathbf{x}_\eta^*),$$

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

  instead of having $(\mathsf{mpk}_2, \mathsf{msk}_2) \leftarrow \mathsf{Setup}_{\mathsf{ext}}^*(1^\lambda, 1^n, 1^{n'})$ and computing

$$\mathsf{ct}_{\eta,2} \leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}_\eta^*, \mathbf{z}_\eta^* - \mathbf{z}_\eta' \| \mathbf{w}_\eta - \mathbf{w}_\eta')), \mathsf{sk}_{f_j,2} \leftarrow \mathsf{KeyGen}_{\mathsf{ext}}(\mathsf{msk}_2, (f_j, [\mathbf{r}]_2))$$

  The simulation-based security of Π_{ext} ensures that $\mathsf{Game}_{\eta.0} \approx_c \mathsf{Game}_{\eta.1}$. The proof is analogous to that of Lemma 9.
- in $\mathsf{Game}_{\eta.2}$, we compute

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

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

  We have $\mathsf{Game}_{\eta.1} \approx_s \mathsf{Game}_{\eta.2}$. This follows from (27) as in Section 7. Here we crucially use the fact that $\mathbf{w}_\eta$ is not leaked to the adversary; the statement (27) does not hold for $\mathbf{w}_i$ with $i \in S$.
- in $\mathsf{Game}_{\eta.3}$, we have $(\mathsf{mpk}_2, \mathsf{msk}_2) \leftarrow \mathsf{Setup}_{\mathsf{ext}}^*(1^\lambda, 1^n, 1^{n'})$ and compute

$$\mathsf{ct}_{\eta,2} \leftarrow \mathsf{Enc}_{\mathsf{ext}}(\mathsf{mpk}_2, (\mathbf{x}_\eta^*, -\mathbf{z}_\eta' \| \mathbf{w}_\eta - \mathbf{w}_\eta')), \mathsf{sk}_{f_j,2} \leftarrow \mathsf{KeyGen}_{\mathsf{ext}}(\mathsf{msk}_2, (f_j, [\mathbf{r}]_2))$$

  We have $\mathsf{Game}_{\eta.2} \approx \mathsf{Game}_{\eta.3}$ and the proof is analogous to that for $\mathsf{Game}_{\eta.1} \approx \mathsf{Game}_{\eta.2}$ from the simulation-based security of Π_{ext}.

Note that, in the proof, the reduction always knows $(\mathbf{x}_i^*, \mathbf{z}_i', \mathbf{w}_i')_{i \in [N]}$ and can simulate $(\mathsf{ct}_{i^*,2}, (\mathsf{ct}_{i,1})_{i \in \bar{S} \setminus \{i^*\}})$ using $(\mathsf{mpk}_1, \mathsf{mpk}_2)$.

Game	$\mathsf{ct}_{i^*,1}$ $\mathsf{ct}_{i^*,2}$	$\mathsf{ct}_{i,1},\ i\in\bar{S}\setminus\{i^*\}, i<\eta$ $\mathsf{ct}_{i,2}$	$\mathsf{ct}_{\eta,1}$ $\mathsf{ct}_{\eta,2}$	$\mathsf{ct}_{i,1},\ i\in\bar{S}\setminus\{i^*\}, i>\eta$ $\mathsf{ct}_{i,2}$	$\mathsf{sk}_{f,1}$ $\mathsf{sk}_{f,2}$	**Remark**
0	real: $\mathbf{x}^*_{i^*}, \mathbf{z}'_{i^*} \| \mathbf{w}'_{i^*}$ real: $\mathbf{x}^*_{i^*}, \mathbf{z}^*_{i^*} - \mathbf{z}'_{i^*} \| \mathbf{w}_{i^*} - \mathbf{w}'_{i^*}$	real: $\mathbf{x}^*_i, \mathbf{z}'_i \| \mathbf{w}'_i$ real: $\mathbf{x}^*_i, \mathbf{z}^*_i - \mathbf{z}'_i \| \mathbf{w}_i - \mathbf{w}'_i$			real: real:	Real game
$0'$	real: $\mathbf{x}^*_{i^*}, \mathbf{z}^*_{i^*} - \mathbf{z}'_{i^*} \| \mathbf{w}_{i^*} - \mathbf{w}'_{i^*}$ real: $\mathbf{x}^*_{i^*}, \mathbf{z}'_{i^*} \| \mathbf{w}'_{i^*}$	real: $\mathbf{x}^*_i, \mathbf{z}'_i \| \mathbf{w}'_i$ real: $\mathbf{x}^*_i, \mathbf{z}^*_i - \mathbf{z}'_i \| \mathbf{w}_i - \mathbf{w}'_i$			real: real:	change of variables
i^*	sim: $\mathbf{x}^*_{i^*}$ real: $\mathbf{x}^*_{i^*}, \mathbf{z}'_{i^*} \| \mathbf{w}'_{i^*}$	real: $\mathbf{x}^*_i, \mathbf{z}'_i \| \mathbf{w}'_i$ real: $\mathbf{x}^*_i, \mathbf{z}^*_i - \mathbf{z}'_i \| \mathbf{w}_i - \mathbf{w}'_i$			sim: $[f(\mathbf{x}^*_{i^*})^\top \mathbf{z}^*_{i^*} - \delta' - \sum \mathbf{w}_i^\top \mathbf{r}]_2$ real:	$\delta' = f(\mathbf{x}^*_{i^*})^\top \mathbf{z}'_{i^*} + (\mathbf{w}'_{i^*})^\top \mathbf{r}$ Π_{ext}
$\eta.0$	sim: $\mathbf{x}^*_{i^*}$ real: $\mathbf{x}^*_{i^*}, \mathbf{z}'_{i^*} \| \mathbf{w}'_{i^*}$	real: $\mathbf{x}^*_i, \mathbf{z}'_i \| \mathbf{w}'_i$ real: $\mathbf{x}^*_i, -\mathbf{z}'_i \| \mathbf{w}_i - \mathbf{w}'_i$	real: $\mathbf{x}^*_\eta, \mathbf{z}'_\eta \| \mathbf{w}'_\eta$ real: $\mathbf{x}^*_\eta, \mathbf{z}^*_\eta - \mathbf{z}'_\eta \| \mathbf{w}_\eta - \mathbf{w}'_\eta$	real: $\mathbf{x}^*_i, \mathbf{z}'_i \| \mathbf{w}'_i$ real: $\mathbf{x}^*_i, \mathbf{z}^*_i - \mathbf{z}'_i \| \mathbf{w}_i - \mathbf{w}'_i$	sim: $[\sum_{i\notin S, i<\eta} f(\mathbf{x}^*_i)^\top \mathbf{z}^*_i - \delta' - \sum \mathbf{w}_i^\top \mathbf{r}]_2$ real:	
$\eta.1$	sim: $\mathbf{x}^*_{i^*}$ real: $\mathbf{x}^*_{i^*}, \mathbf{z}'_{i^*} \| \mathbf{w}'_{i^*}$	real: $\mathbf{x}^*_i, \mathbf{z}'_i \| \mathbf{w}'_i$ real: $\mathbf{x}^*_i, -\mathbf{z}'_i \| \mathbf{w}_i - \mathbf{w}'_i$	real: $\mathbf{x}^*_\eta, \mathbf{z}'_\eta \| \mathbf{w}'_\eta$ sim: $\mathbf{x}^*_\eta$	real: $\mathbf{x}^*_i, \mathbf{z}'_i \| \mathbf{w}'_i$ real: $\mathbf{x}^*_i, \mathbf{z}^*_i - \mathbf{z}'_i \| \mathbf{w}_i - \mathbf{w}'_i$	sim: $[\sum_{i\notin S, i<\eta} f(\mathbf{x}^*_i)^\top \mathbf{z}^*_i - \delta' - \sum \mathbf{w}_i^\top \mathbf{r}]_2$ sim: $[f(\mathbf{x}^*_\eta)^\top \mathbf{z}^*_\eta - f(\mathbf{x}^*_\eta)^\top \mathbf{z}'_\eta - (\mathbf{w}'_\eta)^\top \mathbf{r} + \mathbf{w}_\eta^\top \mathbf{r}]_2$	Π_{ext}
$\eta.2$	sim: $\mathbf{x}^*_{i^*}$ real: $\mathbf{x}^*_{i^*}, \mathbf{z}'_{i^*} \| \mathbf{w}'_{i^*}$	real: $\mathbf{x}^*_i, \mathbf{z}'_i \| \mathbf{w}'_i$ real: $\mathbf{x}^*_i, -\mathbf{z}'_i \| \mathbf{w}_i - \mathbf{w}'_i$	real: $\mathbf{x}^*_\eta, \mathbf{z}'_\eta \| \mathbf{w}'_\eta$ sim: $\mathbf{x}^*_\eta$	real: $\mathbf{x}^*_i, \mathbf{z}'_i \| \mathbf{w}'_i$ real: $\mathbf{x}^*_i, \mathbf{z}^*_i - \mathbf{z}'_i \| \mathbf{w}_i - \mathbf{w}'_i$	sim: $[\sum_{i\notin S, i\leq\eta} f(\mathbf{x}^*_i)^\top \mathbf{z}^*_i - \delta' - \sum \mathbf{w}_i^\top \mathbf{r}]_2$ sim: $[-f(\mathbf{x}^*_\eta)^\top \mathbf{z}'_\eta - (\mathbf{w}'_\eta)^\top \mathbf{r} + \mathbf{w}_\eta^\top \mathbf{r}]_2$	MDDH
$\eta.3$	sim: $\mathbf{x}^*_{i^*}$ real: $\mathbf{x}^*_{i^*}, \mathbf{z}'_{i^*} \| \mathbf{w}'_{i^*}$	real: $\mathbf{x}^*_i, \mathbf{z}'_i \| \mathbf{w}'_i$ real: $\mathbf{x}^*_i, -\mathbf{z}'_i \| \mathbf{w}_i - \mathbf{w}'_i$	real: $\mathbf{x}^*_\eta, \mathbf{z}'_\eta \| \mathbf{w}'_\eta$ real: $\mathbf{x}^*_\eta, -\mathbf{z}'_\eta \| \mathbf{w}_\eta - \mathbf{w}'_\eta$	real: $\mathbf{x}^*_i, \mathbf{z}'_i \| \mathbf{w}'_i$ real: $\mathbf{x}^*_i, \mathbf{z}^*_i - \mathbf{z}'_i \| \mathbf{w}_i - \mathbf{w}'_i$	sim: $[\sum_{i\notin S, i\leq\eta} f(\mathbf{x}^*_i)^\top \mathbf{z}^*_i - \delta' - \sum \mathbf{w}_i^\top \mathbf{r}]_2$ real:	Π_{ext}
$\max\{\bar{S}\}.3$	sim: $\mathbf{x}^*_{i^*}$ real: $\mathbf{x}^*_{i^*}, \mathbf{z}'_{i^*} \| \mathbf{w}'_{i^*}$	real: $\mathbf{x}^*_i, \mathbf{z}'_i \| \mathbf{w}'_i$ real: $\mathbf{x}^*_i, -\mathbf{z}'_i \| \mathbf{w}_i - \mathbf{w}'_i$			sim: $[\sum_{i\notin S} f(\mathbf{x}^*_i)^\top \mathbf{z}^*_i - \delta' - \sum \mathbf{w}_i^\top \mathbf{r}]_2$ real:	$\delta' = f(\mathbf{x}^*_{i^*})^\top \mathbf{z}'_{i^*} + (\mathbf{w}'_{i^*})^\top \mathbf{r}$ Simulator

Fig. 5. Game sequence for Π_{mcl} over $\eta \in \bar{S}\setminus\{i^*\}$ in an ascending order where $i^* = \min\{\bar{S}\}$. Each cell has two rows corresponding to $\mathsf{ct}_{i,1}$ and $\mathsf{ct}_{i,2}$, respectively. Each row is in the format "xxx:yyy" where xxx $\in$ {real, 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 $i \in [N]\setminus\{i^*\}$.

References

1. M. Abdalla, F. Benhamouda, and R. Gay. From single-input to multi-client inner-product functional encryption. In S. D. Galbraith and S. Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 552–582. Springer, Heidelberg, Dec. 2019.
2. M. Abdalla, F. Benhamouda, M. Kohlweiss, and H. Waldner. Decentralizing inner-product functional encryption. In D. Lin and K. Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 128–157. Springer, Heidelberg, Apr. 2019.
3. M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval. Simple functional encryption schemes for inner products. In J. Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 733–751. Springer, Heidelberg, Mar. / Apr. 2015.
4. M. Abdalla, D. Catalano, R. Gay, and B. Ursu. Inner-product functional encryption with fine-grained access control. Cryptology ePrint Archive, Report 2020/577, 2020.
5. M. Abdalla, R. Gay, M. Raykova, and H. Wee. Multi-input inner-product functional encryption from pairings. In J. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 601–626. Springer, Heidelberg, Apr. / May 2017.
6. S. Agrawal, S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption: New perspectives and lower bounds. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 500–518. Springer, Heidelberg, Aug. 2013.
7. S. Agrawal, B. Libert, M. Maitra, and R. Titiu. Adaptive simulation security for inner product functional encryption. In A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 34–64. Springer, Heidelberg, May 2020.
8. S. Agrawal, B. Libert, and D. Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 333–362. Springer, Heidelberg, Aug. 2016.
9. S. Agrawal, M. Maitra, and S. Yamada. Attribute based encryption (and more) for nondeterministic finite automata from LWE. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 765–797. Springer, Heidelberg, Aug. 2019.
10. S. Agrawal, M. Maitra, and S. Yamada. Attribute based encryption for deterministic finite automata from DLIN. In D. Hofheinz and A. Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 91–117. Springer, Heidelberg, Dec. 2019.
11. M. Ambrona, G. Barthe, R. Gay, and H. Wee. Attribute-based encryption in the generic group model: Automated proofs and new constructions. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *ACM CCS 2017*, pages 647–664. ACM Press, Oct. / Nov. 2017.
12. C. E. Z. Baltico, D. Catalano, D. Fiore, and R. Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 67–98. Springer, Heidelberg, Aug. 2017.
13. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988.
14. D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, and D. Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Heidelberg, May 2014.
15. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, Mar. 2011.
16. Z. Brakerski and V. Vaikuntanathan. Circuit-ABE from LWE: Unbounded attributes and semi-adaptive security. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 363–384. Springer, Heidelberg, Aug. 2016.
17. J. Chen, R. Gay, and H. Wee. Improved dual system ABE in prime-order groups via predicate encodings. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 595–624. Springer, Heidelberg, Apr. 2015.
18. J. Chen and H. Wee. Semi-adaptive attribute-based encryption and improved delegation for Boolean formula. In M. Abdalla and R. D. Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 277–297. Springer, Heidelberg, Sept. 2014.
19. Y. Chen, L. Zhang, and S.-M. Yiu. Practical attribute based inner product functional encryption from simple assumptions. Cryptology ePrint Archive, Report 2019/846, 2019.
20. J. Chotard, E. Dufour Sans, R. Gay, D. H. Phan, and D. Pointcheval. Decentralized multi-client functional encryption for inner product. In T. Peyrin and S. Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 703–732. Springer, Heidelberg, Dec. 2018.
21. P. Datta, T. Okamoto, and K. Takashima. Adaptively simulation-secure attribute-hiding predicate encryption. In T. Peyrin and S. Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 640–672. Springer, Heidelberg, Dec. 2018.
22. A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. Villar. An algebraic framework for Diffie-Hellman assumptions. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, Aug. 2013.

23. S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, and H.-S. Zhou. Multi-input functional encryption. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 578–602. Springer, Heidelberg, May 2014.
24. J. Gong, B. Waters, and H. Wee. ABE for DFA from k-lin. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 732–764. Springer, Heidelberg, Aug. 2019.
25. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *45th ACM STOC*, pages 545–554. ACM Press, June 2013.
26. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Predicate encryption for circuits from LWE. In R. Gennaro and M. J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 503–523. Springer, Heidelberg, Aug. 2015.
27. R. Goyal, V. Koppula, and B. Waters. Semi-adaptive security and bundling functionalities made generic and easy. In M. Hirt and A. D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 361–388. Springer, Heidelberg, Oct. / Nov. 2016.
28. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In A. Juels, R. N. Wright, and S. De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 89–98. ACM Press, Oct. / Nov. 2006. Available as Cryptology ePrint Archive Report 2006/309.
29. Y. Ishai and E. Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In P. Widmayer, F. T. Ruiz, R. M. Bueno, M. Hennessy, S. Eidenbenz, and R. Conejo, editors, *ICALP 2002*, volume 2380 of *LNCS*, pages 244–256. Springer, Heidelberg, July 2002.
30. Y. Ishai and H. Wee. Partial garbling schemes and their applications. In J. Esparza, P. Fraigniaud, T. Husfeldt, and E. Koutsoupias, editors, *ICALP 2014, Part I*, volume 8572 of *LNCS*, pages 650–662. Springer, Heidelberg, July 2014.
31. A. Jain, H. Lin, C. Matt, and A. Sahai. How to leverage hardness of constant-degree expanding polynomials overa $\mathbb{R}$ to build $i\mathcal{O}$. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 251–281. Springer, Heidelberg, May 2019.
32. A. Jain, H. Lin, and A. Sahai. Simplifying constructions and assumptions for $i\mathcal{O}$. *IACR Cryptology ePrint Archive*, 2019:1252, 2019.
33. J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, Heidelberg, Apr. 2008.
34. L. Kowalczyk and H. Wee. Compact adaptively secure ABE for NC^1 from k-lin. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 3–33. Springer, Heidelberg, May 2019.
35. A. B. Lewko and B. Waters. Unbounded HIBE and attribute-based encryption. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 547–567. Springer, Heidelberg, May 2011.
36. H. Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 599–629. Springer, Heidelberg, Aug. 2017.
37. T. Okamoto and K. Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 591–608. Springer, Heidelberg, Apr. 2012.
38. T. Okamoto and K. Takashima. Efficient (hierarchical) inner-product encryption tightly reduced from the decisional linear assumption. *IEICE Transactions*, 96-A(1):42–52, 2013.
39. A. Sahai and B. R. Waters. Fuzzy identity-based encryption. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005.
40. B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, Heidelberg, Aug. 2009.
41. B. Waters. Functional encryption for regular languages. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 218–235. Springer, Heidelberg, Aug. 2012.
42. H. Wee. Dual system encryption via predicate encodings. In Y. Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 616–637. Springer, Heidelberg, Feb. 2014.
43. H. Wee. Attribute-hiding predicate encryption in bilinear groups, revisited. In Y. Kalai and L. Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 206–233. Springer, Heidelberg, Nov. 2017.

A Partial Garbling

The following is a restatement of the partial garbling schemes in [30,43] using our notation.

Basic lemma. We review the following lemma for $f(\mathbf{x})^\top \mathbf{z}$ with $f \in \mathcal{F}_{\mathsf{ABP},n,n'}$ which is a subclass of ABP; this shows that we can use the computation of determinant to model the computation of $f(\mathbf{x})^\top \mathbf{z}$.

Lemma 12 ([29,30]). *Given $f \in \mathcal{F}_{\mathsf{ABP},n,n'}$ and $(\mathbf{x},\mathbf{z}) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'}$, we can efficiently compute a matrix $\mathbf{L}_{\mathbf{x},\mathbf{z}} \in \mathbb{Z}_p^{(m+n')\times(m+n')}$ over $\mathbb{Z}_p$ with m depending on f such that*

- $\det \mathbf{L}_{\mathbf{x},\mathbf{z}} = f(\mathbf{x})^\top \mathbf{z}$;
- *each entry of $\mathbf{L}_{\mathbf{x},\mathbf{z}}$ is an affine function in $\mathbf{x},\mathbf{z}$;*
- *$\mathbf{L}_{\mathbf{x},\mathbf{z}}$ contains only -1's in the second diagonal and 0's below the second diagonal.*

Specifically, $\mathbf{L}_{\mathbf{x},\mathbf{z}}$ is obtained by removing the column corresponding to v_0 and the row corresponding to v_1 in the matrix $\mathbf{A}-\mathbf{I}$ where $\mathbf{A}$ is the adjacency matrix of ABP.

In particular, the matrix $\mathbf{L}_{\mathbf{x},\mathbf{z}}$ has the following form where we have an affine function in $\mathbf{x}$ for each $*$:

$$\mathbf{L}_{\mathbf{x},\mathbf{z}} = \left(\begin{array}{ccc|ccc|c} * & * & * & * & * & * & \\ -1 & * & * & * & * & * & \\ & \ddots & * & * & * & * & \\ & & -1 & * & * & * & \\ \hline & & & -1 & & & z_1 \\ & & & & \ddots & & \vdots \\ & & & & & -1 & z_{n'} \end{array}\right)$$

We use $\mathbf{L}_{\mathbf{x}}$ to denote the matrix obtained by removing the last column from $\mathbf{L}_{\mathbf{x},\mathbf{z}}$. Note that $\mathbf{L}_{\mathbf{x}}$ only depends on $\mathbf{x}$ and $f \in \mathcal{F}_{\mathsf{ABP},n,n'}$.

Partial Garbling Scheme. We now review the partial garbling scheme [30,43] for $f(\mathbf{x})^\top \mathbf{z}$. We first introduce a family of matrices of dimension $m+n'$:

$$\mathcal{T} = \left\{ \begin{pmatrix} \mathbf{I}_{m+n'-1} & \mathbf{t} \\ & 1 \end{pmatrix} : \mathbf{t} \in \mathbb{Z}_p^{m+n'-1} \right\}$$

and state the following properties:

1. $\forall \mathbf{T} \in \mathcal{T}$, we have $\det(\mathbf{T}) = 1$;
2. $\forall \mathbf{T} \in \mathcal{T}$, we have $\{\mathbf{T}\binom{\mathbf{t}}{1} : \mathbf{t} \leftarrow \mathbb{Z}_p^{m+n'-1}\} \equiv_s \{\binom{\mathbf{t}}{1} : \mathbf{t} \leftarrow \mathbb{Z}_p^{m+n'-1}\}$;
3. $\exists \mathbf{T} \in \mathcal{T}$ such that $\mathbf{L}_{\mathbf{x},\mathbf{z}}\mathbf{T} = (\mathbf{L}_{\mathbf{x}} | \mathbf{e}_1 \cdot f(\mathbf{x})^\top \mathbf{z})$ for $\mathbf{L}_{\mathbf{x},\mathbf{z}}$ shown above.

The partial garbling scheme is as follows:

- partial garbling for $f,\mathbf{x},\mathbf{z}$:
$$\mathbf{p}_{f,\mathbf{x},\mathbf{z}} = \mathbf{L}_{\mathbf{x},\mathbf{z}} \binom{\mathbf{t}}{1} \tag{28}$$
with random coin $\mathbf{t} \leftarrow \mathbb{Z}_p^{m+n'-1}$.
- reconstruction: we can recover $f(\mathbf{x})^\top \mathbf{z}$ by computing $\det(\mathbf{L}_{\mathbf{x}} | \mathbf{p}_{f,\mathbf{x},\mathbf{z}})$ since
$$\det(\mathbf{L}_{\mathbf{x}} | \mathbf{p}_{f,\mathbf{x},\mathbf{z}}) = \det(\mathbf{L}_{\mathbf{x},\mathbf{z}}\mathbf{T}) = \det \mathbf{L}_{\mathbf{x},\mathbf{z}} = f(\mathbf{x})^\top \mathbf{z} \tag{29}$$
for some $\mathbf{T} \in \mathcal{T}$; here we rely on the property 1 and Lemma 12.
- privacy: given $f(\mathbf{x})^\top \mathbf{z}$, we simulate
$$(\mathbf{L}_{\mathbf{x}} | \mathbf{e}_1 \cdot f(\mathbf{x})^\top \mathbf{z}) \binom{\mathbf{t}}{1}, \quad \mathbf{t} \leftarrow \mathbb{Z}_p^{m+n'-1}. \tag{30}$$
Here we rely on property 2, 3 and Lemma 12.

Algorithm lgen and pgb, pgb*. Let

- $\overline{\mathbf{L}}_{\mathbf{x}} \in \mathbb{Z}_p^{m\times(m+n'-1)}$ be the matrix obtained by removing the last n' rows from $\mathbf{L}_{\mathbf{x}}$;
- $\underline{\mathbf{t}} \in \mathbb{Z}_p^{n'}$ is a vector consisting of the last n' entries of $\mathbf{t}$;

we can rewrite

$$\mathbf{p}_{f,\mathbf{x},\mathbf{z}}^\top = (\mathbf{p}_1^\top, \mathbf{p}_2^\top) = (\mathbf{z}^\top - \underline{\mathbf{t}}^\top, \mathbf{t}^\top \overline{\mathbf{L}}_{\mathbf{x}}^\top). \tag{31}$$

Here we also make a transpose for notational convenience in the context of functional encryptions. Furthermore, since each entry in $\overline{\mathbf{L}}_{\mathbf{x}}^\top$ is an affine function in $\mathbf{x}$, we can write

$$\overline{\mathbf{L}}_{\mathbf{x}}^\top = \mathbf{L}_1(\mathbf{x}\otimes\mathbf{I}_m) + \mathbf{L}_0 \tag{32}$$

where $\mathbf{L}_1$ is formed from n matrices of size $(m+n'-1)\times m$, the ith of which consists of coefficients of x_i in each entries of $\overline{\mathbf{L}}_{\mathbf{x}}^\top$ and $\mathbf{L}_0$ includes the constant term in each entry. Note that $\mathbf{L}_1$ and $\mathbf{L}_0$ only depend on f. In Lemma 2, we let

- lgen output the input-independent matrices $\mathbf{L}_1, \mathbf{L}_0$ on input f;
- pgb, pgb* output (28) and (30) using $\mathbf{L}_1, \mathbf{L}_0$ applying (31) and (32).

Then, it is straightforward to verify the privacy of our extension with shift in Section 4.2 from concrete expressions.

Algorithm rec. Computing $\det(\mathbf{L}_{\mathbf{x}}|\mathbf{p}_{f,\mathbf{x},\mathbf{z}})$ in (29) for reconstruction is computing a linear function in $\mathbf{p}_{f,\mathbf{x},\mathbf{z}}$ by cofactor expansion. In Lemma 2, we let rec output this linear function, represented by a vector $\mathbf{d}_{f,\mathbf{x}}$, and we have $\mathbf{p}_{f,\mathbf{x},\mathbf{z}}^\top \mathbf{d}_{f,\mathbf{x}} = f(\mathbf{x})^\top \mathbf{z}$. Furthermore, it is not hard to see that $(\mathbf{0}, \mathbf{e}_1^\top)\mathbf{d}_{f,\mathbf{x}} = 1$ which is sufficient to verify the reconstruction of our extension with shift in Section 4.2.

B Concrete instantiation of Π_{ubd} in Section 7

We show the concrete instantiation of Π_{ubd} in Section 7 with our concrete scheme Π_{ext} in Section 6.

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

  $$\mathbf{A}_b \leftarrow \mathbb{Z}_p^{(k+1)\times k} \quad \text{and} \quad \mathbf{W}_b \leftarrow \mathbb{Z}_p^{(k+1)\times n'}, \mathbf{U}_b \leftarrow \mathbb{Z}_p^{(k+1)\times kn}, \mathbf{V}_b, \mathbf{W}_{0,b} \leftarrow \mathbb{Z}_p^{(k+1)\times k}, \forall b \in \{1,2\}$$

  and output

  $$\begin{aligned}\mathsf{mpk} &= \left(\mathbb{G}, \{[\mathbf{A}_b^\top]_1, [\mathbf{A}_b^\top\mathbf{W}_b]_1, [\mathbf{A}_b^\top\mathbf{U}_b]_1, [\mathbf{A}_b^\top\mathbf{V}_b]_1, [\mathbf{A}_b^\top\mathbf{W}_{0,b}]_1\}_{b\in\{1,2\}}\right)\\ \mathsf{msk} &= \{\mathbf{W}_b, \mathbf{U}_b, \mathbf{V}_b, \mathbf{W}_{0,b}\}_{b\in\{1,2\}}.\end{aligned}$$

- $\mathsf{Enc}(\mathsf{mpk}, (\mathbf{x}_i, \mathbf{z}_i)_{i\in[N]})$: Sample

  $$\mathbf{w}_2, \ldots, \mathbf{w}_N \leftarrow \mathbb{Z}_p^k \quad \text{and} \quad \mathbf{s}_1, \ldots, \mathbf{s}_N \leftarrow \mathbb{Z}_p^k$$

  and output

  $$\mathsf{ct}_{\{\mathbf{x}_i,\mathbf{z}_i\}} = \begin{pmatrix} \left\{[\mathbf{s}_1^\top\mathbf{A}_1^\top]_1, [\mathbf{z}_1^\top + \mathbf{s}_1^\top\mathbf{A}_1^\top\mathbf{W}_1]_1, [\mathbf{s}_1^\top\mathbf{A}_1^\top\mathbf{U}_1(\mathbf{x}_1\otimes\mathbf{I}_k) + \mathbf{s}_1^\top\mathbf{A}_1^\top\mathbf{V}_1]_1, [-\sum_{i\in[2,N]}\mathbf{w}_i^\top + \mathbf{s}_1^\top\mathbf{A}_1^\top\mathbf{W}_{0,1}]_1\right\} \\ \left\{[\mathbf{s}_i^\top\mathbf{A}_2^\top]_1, [\mathbf{z}_i^\top + \mathbf{s}_i^\top\mathbf{A}_2^\top\mathbf{W}_2]_1, [\mathbf{s}_i^\top\mathbf{A}_2^\top\mathbf{U}_2(\mathbf{x}_i\otimes\mathbf{I}_k) + \mathbf{s}_i^\top\mathbf{A}_2^\top\mathbf{V}_2]_1, [\mathbf{w}_i^\top + \mathbf{s}_i^\top\mathbf{A}_2^\top\mathbf{W}_{0,2}]_1\right\}_{i\in[2,N]} \end{pmatrix} \quad \text{and} \quad (\mathbf{x}_i)_{i\in[N]}.$$

- $\mathsf{KeyGen}(\mathsf{msk}, f)$: 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. Section 4.2). Sample $\mathbf{T}_b \leftarrow \mathbb{Z}_p^{(k+1)\times(m+n'-1)}, \mathbf{R}_b \leftarrow \mathbb{Z}_p^{k\times m}$ for $b \in \{1,2\}$ and $\mathbf{r} \leftarrow \mathbb{Z}_p^k$ and output

  $$\mathsf{sk}_f = \left(\{[\underline{\mathbf{T}}_b + \mathbf{W}_b]_2, [\mathbf{T}_b\mathbf{L}_1 + \mathbf{U}_b(\mathbf{I}_n\otimes\mathbf{R}_b)]_2, [\mathbf{T}_b\mathbf{L}_0 - \mathbf{W}_{0,b}\mathbf{r}\cdot\mathbf{e}_1^\top + \mathbf{V}_b\mathbf{R}_b]_2, [\mathbf{R}_b]_2\}_{b\in\{1,2\}}, [\mathbf{r}]_2\right) \quad \text{and} \quad f$$

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

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

$$\mathsf{sk}_f = \left(\{[\mathbf{K}_{1,b}]_2, [\mathbf{K}_{2,b}]_2, [\mathbf{K}_{3,b}]_2, [\mathbf{R}_b]_2\}_{b\in\{1,2\}}, [\mathbf{r}]_2\right) \quad \text{and} \quad f$$

and ciphertext:

$$\mathsf{ct}_{(\mathbf{x}_i,\mathbf{z}_i)} = \left(\{[\mathbf{c}_{0,i}^\top]_1, [\mathbf{c}_{1,i}^\top]_1, [\mathbf{c}_{2,i}^\top]_1, [\mathbf{c}_{3,i}^\top]_1\}_{i\in[N]}\right) \quad \text{and} \quad (\mathbf{x}_i)_{i\in[N]}$$

the decryption works as follows:

1. compute

$$[\mathbf{p}_{1,1}^\top]_T = e([\mathbf{c}_{1,1}^\top]_1, [\mathbf{I}_{n'}]_2) \cdot e([\mathbf{c}_{0,1}^\top]_1, [-\mathbf{K}_{1,1}]_2)$$

$$[\mathbf{p}_{2,1}^\top]_T = e([\mathbf{c}_{0,1}^\top]_1, [\mathbf{K}_{2,1}(\mathbf{x}_1 \otimes \mathbf{I}_m) + \mathbf{K}_{3,1}]_2) \cdot e([-\mathbf{c}_{2,1}^\top]_1, [\mathbf{R}_1]_2) \cdot e([\mathbf{c}_{3,1}^\top]_1, [\mathbf{r}\cdot\mathbf{e}_1^\top]_2)$$

2. for all $i \in [2, N]$, compute

$$[\mathbf{p}_{1,i}^\top]_T = e([\mathbf{c}_{1,i}^\top]_1, [\mathbf{I}_{n'}]_2) \cdot e([\mathbf{c}_{0,i}^\top]_1, [-\mathbf{K}_{1,2}]_2)$$

$$[\mathbf{p}_{2,i}^\top]_T = e([\mathbf{c}_{0,i}^\top]_1, [\mathbf{K}_{2,2}(\mathbf{x}_i \otimes \mathbf{I}_m) + \mathbf{K}_{3,2}]_2) \cdot e([-\mathbf{c}_{2,i}^\top]_1, [\mathbf{R}_2]_2) \cdot e([\mathbf{c}_{3,i}^\top]_1, [\mathbf{r}\cdot\mathbf{e}_1^\top]_2)$$

3. for all $i \in [N]$, run $\mathbf{d}_{f,\mathbf{x}_i} \leftarrow \mathsf{rec}(f, \mathbf{x}_i)$ (see Section 4.2), compute

$$[D_i]_T = [(\mathbf{p}_{1,i}^\top, \mathbf{p}_{2,i}^\top)\mathbf{d}_{f,\mathbf{x}_i}]_T$$

4. compute

$$[D]_T = [D_1]_T \cdots [D_N]_T$$

and output D via brute-force discrete log.