Functional Encryption for Quadratic Functions from k-Lin, Revisited

Hoeteck Wee

NTT Research

Abstract. We present simple and improved constructions of public-key functional encryption (FE) schemes for quadratic functions. Our main results are:

– an FE scheme for quadratic functions with constant-size keys as well as shorter ciphertexts than all prior schemes based on static assumptions;
– a public-key partially-hiding FE that supports NC1 computation on public attributes and quadratic computation on the private message, with ciphertext size independent of the length of the public attribute.

Both constructions achieve selective, simulation-based security against unbounded collusions, and rely on the (bilateral) k-linear assumption in prime-order bilinear groups. At the core of these constructions is a new reduction from FE for quadratic functions to FE for linear functions.

1 Introduction

In this work, we study functional encryption for quadratic functions. That is, we would like to encrypt a message $\mathbf{z}$ to produce a ciphertext ct, and generate secret keys sk_f for quadratic functions f, so that decrypting ct with sk_f returns $f(\mathbf{z})$ while leaking no additional information about $\mathbf{z}$. In addition, we want (i) short ciphertexts that grow linearly with the length of $\mathbf{z}$, as well as (ii) simulation-based security against collusions, so that an adversary holding ct and secret keys for different functions $f_1, f_2, \ldots$ learns nothing about $\mathbf{z}$ beyond the outputs of these functions. Functional encryption for quadratic functions have a number of applications, including traitor-tracing schemes whose ciphertext size is sublinear in the total number of users [8,6,11,5,16]; obfuscation from simple assumptions [4,18,19,13,20]; as well as privacy-preserving machine learning for neural networks with quadratic activation functions [22].

1.1 Our Results

We present new pairing-based public-key functional encryption (FE) schemes for quadratic functions, improving upon the recent constructions in [21,5,19,12]. Our main results are:

– A FE scheme for quadratic functions with constant-size keys, whose ciphertext size is shorter than those of all prior public-key schemes based on static assumptions [5,12]; moreover, when instantiated over the BLS12-381 curve where $|\mathbb{G}_2| = 2|\mathbb{G}_1|$, our ciphertext size basically matches that of the most efficient scheme in the generic group model [22] (see Fig 1).
– A partially-hiding FE that supports NC1 computation on public attributes $\mathbf{x}$ and quadratic computation on the private message $\mathbf{z}$; moreover, the ciphertext size grows linearly with $\mathbf{z}$ and independent of $\mathbf{x}$. The previous constructions in [13,4,18,19] have ciphertext sizes that grow linearly with both $\mathbf{z}$ and $\mathbf{x}$.

Both constructions achieve selective[1], simulation-based security against unbounded collusions, and rely on the bilateral k-linear assumption in prime-order bilinear groups.

At the core of these constructions is a new reduction from public-key FE for quadratic functions to that for linear functions. The reduction relies on the (bilateral) k-Lin assumption, and blows up the input size by a factor k. Note that the trivial reduction blows up the input size by $|\mathbf{z}|$. Our reduction is simpler and more direct than the previous reductions due to Lin [21] and Gay [12]: (i) we do not require function-hiding FE for linear functions, and (ii) our reduction works directly in the public-key setting. Thanks to (i), we can also decrease the secret key size from linear to constant.

[1] We actually achieve semi-adaptive security [9], a slight strengthening of selective security.

Scheme	$\|\mathsf{ct}\|$	$\|\mathsf{sk}\|$	**Assumption**	**Security**
BCFG17 [5]	$(6n_1+1)\|\mathbb{G}_1\|+(6n_2+1)\|\mathbb{G}_2\|$	$\|\mathbb{G}_1\|+\|\mathbb{G}_2\|$	SXDH, 3-PDDH	SEL-IND
RDGBP19 [22]	$(2n_1+1)\|\mathbb{G}_1\|+2n_2\|\mathbb{G}_2\|$	$\|\mathbb{G}_2\|$	GGM	AD-IND
G20 [12]	$(4n_1+2n_2+2)\|\mathbb{G}_1\|+n_2\|\mathbb{G}_2\|$	$(3n_1+2n_2+2)\|\mathbb{G}_2\|$	SXDH, bi-2-Lin	SA-SIM
GQ20 [14]	$(2n_1+5)\|\mathbb{G}_1\|+(2n_2+5)\|\mathbb{G}_2\|$	$5\|\mathbb{G}_1\|+5\|\mathbb{G}_2\|$	SXDH, bi-2-Lin	SA-SIM
this work	$((k+1)n_1+kn_2+k+1)\|\mathbb{G}_1\|+n_2\|\mathbb{G}_2\|$	$(k+1)\|\mathbb{G}_2\|$	bi-k-Lin, $k>1$	SA-SIM
	$(2n_1+2n_2+2)\|\mathbb{G}_1\|+n_2\|\mathbb{G}_2\|$	$2\|\mathbb{G}_2\|$	SXDH, bi-2-Lin	SA-SIM

Fig. 1. Comparison with prior public-key functional encryption schemes for quadratic functions $f:\mathbb{Z}_p^{n_1}\times\mathbb{Z}_p^{n_2}\to\mathbb{Z}_p$, as well as a concurrent work [14]. Note that $|\mathsf{sk}|$ ignores the contribution from the function f, which is "public". Here, SXDH=1-Lin, and bi-k-Lin (bilateral k-Lin) is a strengthening of k-Lin. 3-PDDH asserts that $[abc]_2$ is pseudorandom given $[a]_1,[b]_2,[c]_1,[c]_2$. In bilinear groups where $|\mathbb{G}|_2=2|\mathbb{G}_1|$, we achieve $|\mathsf{ct}|=(2n_1+4n_2+2)|\mathbb{G}_1|$ under SXDH, bi 2-Lin, almost matching $|\mathsf{ct}|=(2n_1+4n_2+1)|\mathbb{G}_1|$ in RDGBP19.

1.2 Technical Overview

We proceed to provide an overview of our constructions. 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 $\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T$ [10]. We use bold-face lower case to denote row vectors. The k-Lin assumption in $\mathbb{G}_b$ asserts that

$$([\mathbf{A}]_b,[\mathbf{sA}]_b)\approx_c([\mathbf{A}]_b,[\mathbf{u}]_b),\quad \mathbf{s}\leftarrow\mathbb{Z}_p^k,\mathbf{A}\leftarrow\mathbb{Z}_p^{k\times\ell},\mathbf{u}\leftarrow\mathbb{Z}_p^\ell,\ell>k$$

The bilateral k-Lin assumption is a strengthening of k-Lin, and asserts that

$$([\mathbf{A}]_1,[\mathbf{sA}]_1,[\mathbf{A}]_2,[\mathbf{sA}]_2)\approx_c([\mathbf{A}]_1,[\mathbf{u}]_1,[\mathbf{A}]_2,[\mathbf{u}]_2)$$

Note that bilateral 1-Lin is false, for the same reason DDH is false in symmetric bilinear groups.

FE for quadratic functions. Consider the class of quadratic functions over $\mathbb{Z}_p^n\times\mathbb{Z}_p^n$ given by

$$(\mathbf{z}_1,\mathbf{z}_2)\mapsto(\mathbf{z}_1\otimes\mathbf{z}_2)\mathbf{f}^\top$$

where $\mathbf{f}\in\mathbb{Z}_p^{n^2}$ is the coefficient vector. We will first mask $\mathbf{z}_1,\mathbf{z}_2$ in the ciphertext using:

$$[\mathbf{s}_1\mathbf{A}_1+\mathbf{z}_1]_1,[\mathbf{s}_2\mathbf{A}_2+\mathbf{z}_2]_2$$

where the matrices

$$[\mathbf{A}_1]_1,[\mathbf{A}_2]_2,\mathbf{A}_1,\mathbf{A}_2\leftarrow\mathbb{Z}_p^{k\times n}$$

are specified in the master public key. Next, observe that

$$((\mathbf{s}_1\mathbf{A}_1+\mathbf{z}_1)\otimes(\mathbf{s}_2\mathbf{A}_2+\mathbf{z}_2))\cdot\mathbf{f}^\top=(\mathbf{z}_1\otimes\mathbf{z}_2)\mathbf{f}^\top+\text{cross terms}\tag{1}$$

Following [12,21], we will express the cross terms as a linear function evaluated on inputs of length $O(kn)$; the key difference in this work is that the linear function can be derived from the master public key and $\mathbf{f}$.

More precisely, we write

$$\begin{aligned}\underbrace{(\mathbf{s}_1\mathbf{A}_1+\mathbf{z}_1)}_{\mathbf{y}_1}\otimes\underbrace{(\mathbf{s}_2\mathbf{A}_2+\mathbf{z}_2)}_{\mathbf{y}_2}&=(\mathbf{z}_1\otimes\mathbf{z}_2)+\mathbf{s}_1\mathbf{A}_1\otimes\mathbf{z}_2 \qquad +\mathbf{y}_1\otimes\mathbf{s}_2\mathbf{A}_2\\&=(\mathbf{z}_1\otimes\mathbf{z}_2)+(\mathbf{s}_1\otimes\mathbf{z}_2)\cdot(\mathbf{A}_1\otimes\mathbf{I}_n)\qquad+(\mathbf{y}_1\otimes\mathbf{s}_2)\cdot(\mathbf{I}_n\otimes\mathbf{A}_2)\\&=(\mathbf{z}_1\otimes\mathbf{z}_2)+(\mathbf{s}_1\otimes\mathbf{z}_2\|\mathbf{y}_1\otimes\mathbf{s}_2)\binom{\mathbf{A}_1\otimes\mathbf{I}_n}{\mathbf{I}_n\otimes\mathbf{A}_2}\end{aligned}$$

where the second equality uses the mixed-product property of the tensor product, which tells us that $(\mathbf{M}_1 \otimes \mathbf{M}_2)(\mathbf{M}_3 \otimes \mathbf{M}_4) = (\mathbf{M}_1\mathbf{M}_3) \otimes (\mathbf{M}_2\mathbf{M}_4)$, and $\|$ denotes row vector concatenation. Multiplying both sides on the right by $\mathbf{f}^\top$ and rearranging the terms yields:

$$(\mathbf{z}_1 \otimes \mathbf{z}_2)\mathbf{f}^\top = (\mathbf{y}_1 \otimes \mathbf{y}_2)\mathbf{f}^\top - \boxed{(\mathbf{s}_1 \otimes \mathbf{z}_2 \| \mathbf{y}_1 \otimes \mathbf{s}_2)\mathbf{M}\mathbf{f}^\top} \tag{2}$$

where $\mathbf{M} := \binom{\mathbf{A}_1 \otimes \mathbf{I}_n}{\mathbf{I}_n \otimes \mathbf{A}_2}$. As we mentioned earlier, the boxed term (= cross terms in (1))

$$(\mathbf{s}_1 \otimes \mathbf{z}_2 \| \mathbf{y}_1 \otimes \mathbf{s}_2) \cdot \mathbf{M}\mathbf{f}^\top \tag{3}$$

corresponds to a linear computation where

– the input $(\mathbf{s}_1 \otimes \mathbf{z}_2 \| \mathbf{y}_1 \otimes \mathbf{s}_2)$ has length $O(kn)$;
– the linear function $\mathbf{M}\mathbf{f}^\top$ can be computed given $\mathbf{f}$ and the matrices $\mathbf{A}_1, \mathbf{A}_2$ in the public key.

The latter property pertaining to $\mathbf{M}\mathbf{f}^\top$ is what allows us to significantly simplify the previous reductions in [21,12], since there is nothing "secret" about the linear function $\mathbf{M}\mathbf{f}^\top$. In the prior works, the linear function leaks information about the master secret key beyond what can be computed from the master public key.

In particular, we can use a public-key FE for linear functions (linear FE for short) [3,23,1] to compute (3). That is, we encrypt $[\mathbf{s}_1 \otimes \mathbf{z}_2 \| \mathbf{y}_1 \otimes \mathbf{s}_2]_1$, and generate a secret key for $[\mathbf{M}\mathbf{f}^\top]_2$. The linear FE schemes in [3,23] extend readily to this setting where both the input and function are specified "in the exponent"; moreover, these schemes achieve selective, simulation-based security under the k-Lin assumption, with constant-size secret keys. The linear FE ciphertext would lie in $\mathbb{G}_1$, whereas both $\mathbf{M}$ and the secret key would lie in $\mathbb{G}_2$. Note that in order to compute $[\mathbf{M}]_2$, we would also publish $[\mathbf{A}_1]_2$ in the public key. We present a self-contained description of our quadratic FE in Section A.

Security overview. Security, intuitively, is fairly straight-forward:

– First, observe that $[\mathbf{y}_1]_1, [\mathbf{y}_2]$ leaks no information about $\mathbf{z}_1, \mathbf{z}_2$, thanks to the k-Lin assumption;
– Next, we can simulate the ciphertext and secret key for the linear FE given $(\mathbf{s}_1 \otimes \mathbf{z}_2 \| \mathbf{y}_1 \otimes \mathbf{s}_2)\mathbf{M}\mathbf{f}^\top$, which we can rewrite as $(\mathbf{z}_1 \otimes \mathbf{z}_2)\mathbf{f}^\top - (\mathbf{y}_1 \otimes \mathbf{y}_2)\mathbf{f}^\top$. We can in turn compute the latter given just $\mathbf{y}_1, \mathbf{y}_2$ and the output of the ideal functionality and therefore the linear FE ciphertext-key pair leaks no additional information about $\mathbf{z}_1, \mathbf{z}_2$.

In the reduction, we would need to compute $[\mathbf{y}_1 \otimes \mathbf{y}_2]_2$ in order to simulate the secret key for the linear FE. This is something we can compute given either $\mathbf{y}_1, [\mathbf{y}_2]_2$ or $[\mathbf{y}_1]_2, \mathbf{y}_2$. The latter along with publishing $[\mathbf{A}_1]_2$ in the public key is why we require the bilateral k-Lin assumption. For the most efficient concrete instantiation, we will use the bilateral 2-Lin assumption together with SXDH (i.e., 1-Lin), where we sample $\mathbf{A}_1 \leftarrow \mathbb{Z}_p^{2 \times n}, \mathbf{A}_2 \leftarrow \mathbb{Z}_p^{1 \times n}$. We leave the question of basing quadratic FE solely on the standard k-Lin assumption as an open problem.

Extension to partially hiding FE. Our approach extends readily to partially hiding FE (PHFE) for the class

$$(\overbrace{\mathbf{x}}^{\text{public}}, \overbrace{(\mathbf{z}_1, \mathbf{z}_2)}^{\text{private}}) \mapsto (\mathbf{z}_1 \otimes \mathbf{z}_2) f(\mathbf{x})^\top$$

where f captures NC1 –more generally, any arithmetic branching program– computation on the public attribute $\mathbf{x}$ and outputs a vector in $\mathbb{Z}_p^{n^2}$. Note that FE for quadratic functions corresponds to the special case where f is a constant function (independent of $\mathbf{x}$). The idea behind the extension to PHFE is to replace $\mathbf{f}^\top$ in (2) with $f(\mathbf{x})$ (the decryptor can compute $f(\mathbf{x})$ since $\mathbf{x}$ is public), which yields:

$$(\mathbf{z}_1 \otimes \mathbf{z}_2) f(\mathbf{x})^\top = (\mathbf{y}_1 \otimes \mathbf{y}_2) f(\mathbf{x})^\top - \boxed{(\mathbf{s}_1 \otimes \mathbf{z}_2 \| \mathbf{y}_1 \otimes \mathbf{s}_2)\mathbf{M} f(\mathbf{x})^\top}$$

To compute the new boxed term, we will rely on the partially-hiding linear FE scheme in [2] for the class

$$(\overbrace{\mathbf{x}}^{\text{public}}, \overbrace{\mathbf{z}}^{\text{private}}) \mapsto \mathbf{z} f(\mathbf{x})^\top$$

We can augment the construction to take into account the matrix $\mathbf{M}$; some care is needed as the decryption algorithm only gets $[\mathbf{M}]_2$ and not $\mathbf{M}$. In the ensuing scheme as with [2], the ciphertext size grows linearly with the message and independent of $\mathbf{x}$, which we then inherit in our partially-hiding quadratic FE.

2 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 *row* vectors and upper case boldcase to denote matrices. We use $\mathbf{e}_i$ to denote the i'th elementary row 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 $\{1, 2, \ldots, N\}$.

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

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

The mixed-product property for tensor product says that

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD})$$

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.

2.1 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 $\mathbb{G}_1$ [10]:

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

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

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.* [10] showed that

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

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

The bilateral MDDH assumption is defined analogously with the advantage function:

$$\big| \Pr[\mathcal{A}(\mathbb{G}, [\mathbf{M}]_1, \boxed{[\mathbf{MS}]_1}, [\mathbf{M}]_2, \boxed{[\mathbf{MS}]_1}) = 2] - \Pr[\mathcal{A}(\mathbb{G}, [\mathbf{M}]_1, [\mathbf{U}]_1, [\mathbf{M}]_2, [\mathbf{U}]_2) = 1] \big|$$

2.2 Partially-Hiding Functional Encryption (PHFE)

We recall the notion of partially-hiding functional encryption [15,23,4,7] for the function class

$$(\mathbf{x},\mathbf{z}) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'} \mapsto h(\mathbf{z}) f(\mathbf{x})^\top$$

where $h : \mathbb{Z}_p^{n'} \to \mathbb{Z}_p^{n''}$ is fixed and $f \in \mathcal{F}_{\mathsf{ABP},n,n''}$ is specified by the secret key. We will be primarily interested in the settings $h(\mathbf{z}) = \mathbf{z}$ and $h(\mathbf{z}_1,\mathbf{z}_2) = \mathbf{z}_1 \otimes \mathbf{z}_2$, which generalize FE for linear functions and quadratic functions respectively.

Syntax. A *partially-hiding functional encryption scheme* (PHFE) consists of four algorithms:

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

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

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

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

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

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 $h(\mathbf{z}) f(\mathbf{x})^\top$ if the value is bounded by B. This limitation is also present in prior works on (IP)FE from DDH and bilinear groups [1,3,21,5], 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 [9] (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 $\mathcal{A}$,

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

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

3 Main Construction

In this section, we present our PHFE scheme for the class

$$(\overbrace{\mathbf{x}}^{\text{public}}, \overbrace{(\mathbf{z}_1,\mathbf{z}_2)}^{\text{private}}) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'_1+n'_2} \mapsto (\mathbf{z}_1 \otimes \mathbf{z}_2) f(\mathbf{x})^\top, \quad f \in \mathcal{F}_{\mathsf{ABP},n,n'_1 n'_2}$$

The scheme is SA-SIM-secure under the bilateral k-Lin assumption and the k'-Lin assumption in $\mathbb{G}_1, \mathbb{G}_2$ (for the most efficient concrete instantiation, we set $k = 2, k' = 1$). In our scheme, decryption actually computes $[(\mathbf{z}_1 \otimes \mathbf{z}_2) f(\mathbf{x})^\top]_T$, whereas the simulator only needs to get $[(\mathbf{z}_1 \otimes \mathbf{z}_2) f(\mathbf{x})^\top]_2$. Note that FE for quadratic functions is a special case of our PHFE (where f has the quadratic function hard-wired into it). We present a self-contained description of our quadratic FE in Section A.

As a building block, we rely on a SA-SIM-secure PHFE scheme $(\mathsf{Setup}_0, \mathsf{Enc}_0, \mathsf{KeyGen}_0, \mathsf{Dec}_0)$ for the class

$$(\overbrace{\mathbf{x}}^{\text{public}}, \overbrace{\mathbf{z}}^{\text{private}}) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{k'n_1'+kn_2'} \mapsto [\mathbf{z}\mathbf{M} f(\mathbf{x})^\top]_T, \quad f \in \mathcal{F}_{\mathsf{ABP},n,n_1'n_2'}$$

parameterized by a matrix $[\mathbf{M}]_2 \in \mathbb{G}_1^{(k'n_1'+kn_2')\times n_1'n_2'}$, where encryption gets $[\mathbf{z}]_1$ and the simulator gets $[\mathbf{z}\mathbf{M} f(\mathbf{x})^\top]_2$. We instantiate the building block in Section 4.

3.1 Our Scheme

- $\mathsf{Setup}(p, 1^n, 1^{n_1'}, 1^{n_2'})$: Run $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(p)$. Sample

$$\mathbf{A}_1 \leftarrow \mathbb{Z}_p^{k\times n_1'}, \mathbf{A}_2 \leftarrow \mathbb{Z}_p^{k'\times n_2'}, (\mathsf{mpk}_0, \mathsf{msk}_0) \leftarrow \mathsf{Setup}_0(p, 1^n, 1^{k'n_1'+kn_2'}, [\mathbf{M}]_2)$$

  where

$$\mathbf{M} := \begin{pmatrix} \mathbf{A}_1 \otimes \mathbf{I}_{n_2'} \\ \mathbf{I}_{n_1'} \otimes \mathbf{A}_2 \end{pmatrix} \in \mathbb{Z}_p^{(k'n_1'+kn_2')\times n_1'n_2'}$$

  and output

$$\mathsf{mpk} = \big(\mathbb{G}, [\mathbf{A}_1]_1, [\mathbf{A}_1]_2, [\mathbf{A}_2]_2, \mathsf{mpk}_0\big) \quad \text{and} \quad \mathsf{msk} = \mathsf{msk}_0$$

  Observe that given mpk, we can compute $[\mathbf{M}]_2$.
- $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, (\mathbf{z}_1, \mathbf{z}_2))$: Sample

$$\mathbf{s}_1 \leftarrow \mathbb{Z}_p^k, \mathbf{s}_0, \mathbf{s}_2 \leftarrow \mathbb{Z}_p^{k'}, \mathsf{ct}_0 \leftarrow \mathsf{Enc}_0\big(\mathsf{mpk}_0, \mathbf{x}, [\mathbf{s}_1 \otimes \mathbf{z}_2 \| \underbrace{(\mathbf{s}_1\mathbf{A}_1 + \mathbf{z}_1)}_{\mathbf{y}_1} \otimes \mathbf{s}_2]_1\big)$$

  and output

$$\mathsf{ct} = \big([\underbrace{\mathbf{s}_1\mathbf{A}_1 + \mathbf{z}_1}_{\mathbf{y}_1}]_1, [\underbrace{\mathbf{s}_2\mathbf{A}_2 + \mathbf{z}_2}_{\mathbf{y}_2}]_2, \mathsf{ct}_0\big)$$

- $\mathsf{KeyGen}(\mathsf{msk}, f)$: Output

$$\mathsf{sk}_f \leftarrow \mathsf{KeyGen}_0(\mathsf{msk}_0, f)$$

- $\mathsf{Dec}(\mathsf{sk}_f, f, \mathsf{ct}, \mathbf{x})$: Output

$$[(\mathbf{y}_1 \otimes \mathbf{y}_2) \cdot f(\mathbf{x})^\top]_T \cdot \Big(\mathsf{Dec}_0(\mathsf{sk}_f, (f, [\mathbf{M}]_2), \mathsf{ct}_0, \mathbf{x})\Big)^{-1}$$

Correctness. First, observe that we have

$$\begin{aligned} \underbrace{(\mathbf{s}_1\mathbf{A}_1 + \mathbf{z}_1)}_{\mathbf{y}_1} \otimes \underbrace{(\mathbf{s}_2\mathbf{A}_2 + \mathbf{z}_2)}_{\mathbf{y}_2} &= (\mathbf{z}_1 \otimes \mathbf{z}_2) + \mathbf{s}_1\mathbf{A}_1 \otimes \mathbf{z}_2 + \mathbf{y}_1 \otimes \mathbf{s}_2\mathbf{A}_2 \\ &= (\mathbf{z}_1 \otimes \mathbf{z}_2) + (\mathbf{s}_1 \otimes \mathbf{z}_2)\cdot(\mathbf{A}_1 \otimes \mathbf{I}_{n_2'}) + (\mathbf{y}_1 \otimes \mathbf{s}_2)\cdot(\mathbf{I}_{n_1'} \otimes \mathbf{A}_2) \\ &= (\mathbf{z}_1 \otimes \mathbf{z}_2) + (\mathbf{s}_1 \otimes \mathbf{z}_2 \| \mathbf{y}_1 \otimes \mathbf{s}_2)\mathbf{M} \end{aligned} \tag{4}$$

where the second equality uses the mixed-product property of the tensor product. Multiplying both sides of (4) by $f(\mathbf{x})^\top$ and rearranging the terms yields:

$$(\mathbf{z}_1 \otimes \mathbf{z}_2) f(\mathbf{x})^\top = (\mathbf{y}_1 \otimes \mathbf{y}_2) f(\mathbf{x})^\top - (\mathbf{s}_1 \otimes \mathbf{z}_2 \| \mathbf{y}_1 \otimes \mathbf{s}_2)\mathbf{M} f(\mathbf{x})^\top \tag{5}$$

Next, correctness of the underlying scheme tells us that

$$\mathsf{Dec}_0(\mathsf{sk}_f, (f, [\mathbf{M}]_2), \mathsf{ct}_0, \mathbf{x}) = (\mathbf{s}_1 \otimes \mathbf{z}_2 \| \mathbf{y}_1 \otimes \mathbf{s}_2)\mathbf{M} f(\mathbf{x})^\top$$

Correctness then follows readily.

3.2 Simulator

We start by describing the simulator.

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

$$\mathbf{A}_1 \leftarrow \mathbb{Z}_p^{k\times n'_1}, \mathbf{A}_2 \leftarrow \mathbb{Z}_p^{k'\times n'_2}, (\mathsf{mpk}_0^*, \mathsf{msk}_0^*) \leftarrow \mathsf{Setup}_0^*(p, 1^n, 1^{k'n_1+kn_2})$$

  and output

$$\mathsf{mpk}^* = \left(\mathbb{G}, [\mathbf{A}_1]_1, [\mathbf{A}_1]_2, [\mathbf{A}_2]_2, \mathsf{mpk}_0^*\right) \quad \text{and} \quad \mathsf{msk}^* = \mathsf{msk}_0^*$$

- $\mathsf{Enc}^*(\mathsf{msk}_0^*, \mathbf{x}^*)$: Sample

$$\mathbf{y}_1 \leftarrow \mathbb{Z}_p^{n'_1}, \mathbf{y}_2 \leftarrow \mathbb{Z}_p^{n'_2}, \mathsf{ct}_0^* \leftarrow \mathsf{Enc}_0^*(\mathsf{msk}_0^*, \mathbf{x}^*)$$

  and output

$$\mathsf{ct}^* = \left([\mathbf{y}_1]_1, [\mathbf{y}_2]_1, \mathsf{ct}_0^*\right)$$

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

$$\mathsf{sk}_f \leftarrow \mathsf{KeyGen}_0^*(\mathsf{msk}_0^*, \mathbf{x}^*, f, [(\mathbf{y}_1 \otimes \mathbf{y}_2) f(\mathbf{x}^*)]_T \cdot [\mu]_2^{-1})$$

3.3 Proof of Security

We proceed via a series of games and we use Adv_i to denote the advantage of $\mathcal{A}$ in Game i. Let $\mathbf{x}^*, (\mathbf{z}_1^*, \mathbf{z}_2^*)$ denote the semi-adaptive challenge.

Game 0. Real game.

Game 1. Replace $(\mathsf{Setup}_0, \mathsf{Enc}_0, \mathsf{KeyGen}_0)$ in Game_0 with $(\mathsf{Setup}_0^*, \mathsf{Enc}_0^*, \mathsf{KeyGen}_0^*)$ where

$$\begin{aligned} \mathsf{ct}^* &= ([\mathbf{y}_1]_1, [\mathbf{y}_2]_2, \mathsf{Enc}_0(\mathsf{msk}^*, \mathbf{x}^*)), \; \mathbf{y}_1 = \mathbf{s}_1\mathbf{A}_1 + \mathbf{z}_1^*, \mathbf{y}_2 = \mathbf{s}_2\mathbf{A}_2 + \mathbf{z}_2^* \\ \mathsf{sk}_f &\leftarrow \mathsf{KeyGen}_0^*(\mathsf{msk}_0^*, \mathbf{x}^*, f, \boxed{[(\mathbf{s}_1 \otimes \mathbf{z}_2^* \| \mathbf{y}_1 \otimes \mathbf{s}_2)\mathbf{M} f(\mathbf{x}^*)^\top]_2}) \end{aligned}$$

We have $\mathsf{Game}_1 \approx_c \mathsf{Game}_0$, by security of the underlying PHFE scheme. The reduction samples

$$\mathbf{A}_1 \leftarrow \mathbb{Z}_p^{k\times n'_1}, \mathbf{A}_2 \leftarrow \mathbb{Z}_p^{k'\times n'_2}, \mathbf{s}_1 \leftarrow \mathbb{Z}_p^k, \mathbf{s}_0, \mathbf{s}_2 \leftarrow \mathbb{Z}_p^{k'},$$

and upon receiving $\mathbf{x}^*, (\mathbf{z}_1^*, \mathbf{z}_2^*)$ from $\mathcal{A}$, sends

$$\mathbf{x}^*, \mathbf{s}_1 \otimes \mathbf{z}_2^* \| (\mathbf{s}_1\mathbf{A}_1 + \mathbf{z}_1^*) \otimes \mathbf{s}_2$$

as the semi-adaptive challenge.

Game 2. Replace sk_f in Game 1 with

$$\mathsf{sk}_f \leftarrow \mathsf{KeyGen}_0^*(\mathsf{msk}_0^*, \mathbf{x}^*, f, [(\mathbf{y}_1 \otimes \mathbf{y}_2) f(\mathbf{x}^*)^\top]_2 \cdot [(\mathbf{z}_1^* \otimes \mathbf{z}_2^*) f(\mathbf{x}^*)^\top]_2^{-1})$$

Here, we have $\mathsf{Game}_2 \equiv \mathsf{Game}_1$, thanks to (5), which tells us that

$$[(\mathbf{y}_1 \otimes \mathbf{y}_2) f(\mathbf{x}^*)^\top]_2 \cdot [(\mathbf{z}_1^* \otimes \mathbf{z}_2^*) f(\mathbf{x}^*)^\top]_2^{-1} = [(\mathbf{s}_1 \otimes \mathbf{z}_2^* \| \mathbf{y}_1 \otimes \mathbf{s}_2)\mathbf{M} f(\mathbf{x}^*)^\top]_2$$

Game 3. We replace $\boxed{[\mathbf{s}_1\mathbf{A}_1+\mathbf{z}_1^*]_1}$ in ct^* in Game_2 with $[\mathbf{y}_1]_1$ where $\mathbf{y}_1 \leftarrow \mathbb{Z}_q^{n_1'}$. Then, we have $\mathsf{Game}_3 \approx_c \mathsf{Game}_2$ via the bi-lateral k-Lin assumption. The assumption tells us that for all $\mathbf{z}_1^*$,

$$([\mathbf{A}_1]_1,[\mathbf{A}_1]_2,[\mathbf{s}^\top\mathbf{A}_1+\mathbf{z}_1^*]_1,[\mathbf{s}^\top\mathbf{A}_1+\mathbf{z}_1^*]_2) \approx_c ([\mathbf{A}_1]_1,[\mathbf{A}_1]_2,[\mathbf{y}_1]_1,[\mathbf{y}_1]_2)$$

where $\mathbf{s} \leftarrow \mathbb{Z}_p^k, \mathbf{y}_1 \leftarrow \mathbb{Z}_p^{n_1'}$. Note that this holds even if $\mathbf{z}_1^*$ is adaptively chosen after seeing $[\mathbf{A}_1]_1,[\mathbf{A}_1]_2$. The reduction then samples

$$\mathbf{A}_2 \leftarrow \mathbb{Z}_p^{k'\times n_2'},\ \mathbf{s}_2 \leftarrow \mathbb{Z}_p^{k'},\ (\mathsf{mpk}_0^*,\mathsf{msk}_0^*) \leftarrow \mathsf{Setup}_0^*(p,1^n,1^{k'n_1+kn_2})$$

sets $\mathbf{y}_2 := \mathbf{s}_2\mathbf{A}_2+\mathbf{z}_2^*$, and uses the fact that in Games 2 and 3,

- it can compute $\mathsf{mpk}^*,\mathsf{ct}^*$ given $[\mathbf{A}_1]_1,[\mathbf{A}_1]_2,[\mathbf{y}_1]_1$ respectively;
- it can sample sk_f by using $[\mathbf{y}_1]_2,\mathbf{y}_2$ to compute $[\mathbf{y}_1\otimes\mathbf{y}_2]_2$.

Game 4. We replace $\boxed{[\mathbf{s}_2\mathbf{A}_2+\mathbf{z}_2^*]_1}$ in ct^* in Game_3 with $[\mathbf{y}_2]_1$ where $\mathbf{y}_2 \leftarrow \mathbb{Z}_q^{n_2'}$. Then, we have $\mathsf{Game}_4 \approx_c \mathsf{Game}_3$ via the k'-Lin assumption in $\mathbb{G}_2$. Here, we use the fact that we can sample sk_f in Games 3 and 4 using $\mathbf{y}_1,[\mathbf{y}_2]_2$ to compute $[\mathbf{y}_1\otimes\mathbf{y}_2]_2$.

Finally, note that Game_4 is exactly the output of the simulator.

4 Partially-Hiding FE for Linear Functions

In this section, we present our PHFE scheme for the class

$$(\overbrace{\mathbf{x}}^{\text{public}},\overbrace{\mathbf{z}}^{\text{private}}) \mapsto [\mathbf{zM}f(\mathbf{x})^\top]_T$$

parameterized by a matrix $[\mathbf{M}]_2$, where encryption gets $[\mathbf{z}]_1$, and the simulator gets $[\mathbf{zM}f(\mathbf{x})^\top]_2$. In fact, we present a scheme for a more general setting where the matrix $[\mathbf{M}]_2$ is specified by the function corresponding to the secret key (that is, we allow a different $[\mathbf{M}]_2$ for each secret key, rather than the same matrix for all keys). The scheme is a somewhat straight-forward modification of that in [2]; some care is needed as the decryption algorithm only gets $[\mathbf{M}]_2$ and not $\mathbf{M}$. This scheme achieves simulation-based semi-adaptive security under k-Lin. Most of the text in this section is copied verbatim from [2], with minor adaptations to account for $\mathbf{M}$.

4.1 Partial Garbling Scheme

The partial garbling scheme [2,17,23] for $\mathbf{z}f(\mathbf{x})^\top$ 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:

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

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

Lemma 1 (partial garbling [2,17,23]). *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 $\mathbf{L}_0 \in \mathbb{Z}_p^{t\times mn}, \mathbf{L}_1 \in \mathbb{Z}_p^{t\times m}$, and*

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

where $\mathbf{t} \in \mathbb{Z}_p^t$ and $\underline{\mathbf{t}}$ consists of the last n' entries in $\mathbf{t}$ and m,t 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* $f,\mathbf{x},\mathbf{z},\mathbf{t}$,

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

where $\mathbf{p}_{f,\mathbf{x},\mathbf{z}} = \mathsf{pgb}(f,\mathbf{x},\mathbf{z};\mathbf{t})$.

– *(privacy) for all* $f,\mathbf{x},\mathbf{z}$,

$$\mathsf{pgb}(f,\mathbf{x},\mathbf{z};\mathbf{t}) \approx_s \mathsf{pgb}^*(f,\mathbf{x},\mathbf{z}f(\mathbf{x})^{\top};\mathbf{t})$$

where the randomness is over $\mathbf{t} \leftarrow \mathbb{Z}_p^t$.

4.2 Construction

Our scheme Π is similar to Π_{one} in [2], with the modifications marked using boxed terms. We rely on partial garbling to compute $\mathsf{pgb}(f,\mathbf{x},\boxed{\mathbf{zM}};\mathbf{t})$ instead of $\mathsf{pgb}(f,\mathbf{x},\mathbf{z};\mathbf{t})$ "in the exponent" over $\mathbb{G}_T$; applying the reconstruction algorithm (which requires knowing $f,\mathbf{x}$ but not $\mathbf{M}$) then returns $[\boxed{\mathbf{zM}}f(\mathbf{x})^{\top}]_T$.

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

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

and output

$$\mathsf{mpk} = \big(\mathbb{G}, [\mathbf{A}]_1, [\mathbf{AW}]_1, [\mathbf{AU}]_1, [\mathbf{AV}]_1\big) \quad \text{and} \quad \mathsf{msk} = \big(\mathbf{W}, \mathbf{U}, \mathbf{V}\big).$$

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

$$\mathsf{ct}_{\mathbf{x},\mathbf{z}} = \big([\mathbf{sA}]_1, [\mathbf{z}+\mathbf{sAW}]_1, [\mathbf{sAU}(\mathbf{x}^{\top}\otimes\mathbf{I}_k)+\mathbf{sAV}]_1\big) \quad \text{and} \quad \mathbf{x}.$$

Note that it is sufficient for Enc to get $[\mathbf{z}]_1$.

– $\mathsf{KeyGen}(\mathsf{msk},(f,[\mathbf{M}]_2))$: Run $(\mathbf{L}_1,\mathbf{L}_0) \leftarrow \mathsf{lgen}(f)$ where $\mathbf{L}_1 \in \mathbb{Z}_p^{t\times mn}, \mathbf{L}_0 \in \mathbb{Z}_p^{t\times m}$ (cf. Section 4.1). Sample $\mathbf{T} \leftarrow \mathbb{Z}_p^{(k+1)\times t}$ and $\mathbf{R} \leftarrow \mathbb{Z}_p^{k\times m}$ and output

$$\mathsf{sk}_{f,\mathbf{M}} = \big([\underline{\mathbf{T}}+\boxed{\mathbf{WM}}]_2, [\mathbf{TL}_1+\mathbf{U}(\mathbf{I}_n\otimes\mathbf{R})]_2, [\mathbf{TL}_0+\mathbf{VR}]_2, [\mathbf{R}]_2\big) \quad \text{and} \quad (f,[\mathbf{M}]_2).$$

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

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

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

and ciphertext:

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

the decryption works as follows:

1. compute

$$[\mathbf{p}_1]_T = e([\mathbf{c}_1]_1,\boxed{[\mathbf{M}]_2})\cdot e([\mathbf{c}_0]_1,[-\mathbf{K}_1]_2) \tag{6}$$

2. compute

$$[\mathbf{p}_2]_T = e([\mathbf{c}_0]_1,[\mathbf{K}_2(\mathbf{x}^{\top}\otimes\mathbf{I}_m)+\mathbf{K}_3]_2)\cdot e([-\mathbf{c}_2]_1,[\mathbf{R}]_2) \tag{7}$$

3. run $\mathbf{d}_{f,\mathbf{x}} \leftarrow \mathsf{rec}(f,\mathbf{x})$ (cf. Section 4.1), compute

$$[D]_T = [(\mathbf{p}_1\|\mathbf{p}_2)\mathbf{d}_{f,\mathbf{x}}^{\top}]_T \tag{8}$$

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

$$\begin{aligned} \mathbf{p}_1 &= \mathbf{zM} - \mathbf{sA}\underline{\mathbf{T}} && (9)\\ \mathbf{p}_2 &= \mathbf{sATL}_1(\mathbf{x}^\top \otimes \mathbf{I}_m) + \mathbf{sATL}_0 && (10)\\ (\mathbf{p}_1 \| \mathbf{p}_2)\mathbf{d}_{f,\mathbf{x}}^\top &= \mathbf{zM}f(\mathbf{x}) && (11) \end{aligned}$$

Here (11) follows from the fact that

$$(\mathbf{p}_1 \| \mathbf{p}_2) = \mathsf{pgb}(f, \mathbf{x}, \mathbf{zM}; (\mathbf{sAT})) \quad \text{and} \quad \mathbf{d}_{f,\mathbf{x}} = \mathsf{rec}(f, \mathbf{x})$$

and reconstruction of the partial garbling in (6); the remaining two equalities follow from:

$$\begin{aligned} (9) \quad && \mathbf{zM} - \mathbf{sA}\underline{\mathbf{T}} &= (\mathbf{z} + \mathbf{sAW}) \cdot \mathbf{M} - \mathbf{sA} \cdot (\underline{\mathbf{T}} + \mathbf{WM})\\ (10) \quad && \mathbf{sATL}_1(\mathbf{x}^\top \otimes \mathbf{I}_m) + \mathbf{sATL}_0 &= \mathbf{sA} \cdot \big((\mathbf{TL}_1 + \mathbf{U}(\mathbf{I}_n \otimes \mathbf{R}))(\mathbf{x}^\top \otimes \mathbf{I}_m) + (\mathbf{TL}_0 + \mathbf{VR})\big) - \big(\mathbf{sAU}(\mathbf{x}^\top \otimes \mathbf{I}_k) + \mathbf{sAV}\big) \cdot \mathbf{R} \end{aligned}$$

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

Simulator. We describe the simulator. We defer the analysis to Section B.

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

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

  and output

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

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

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

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

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

  Sample $\mathbf{T} \leftarrow \mathbb{Z}_p^{(k+1)\times t}$, $\hat{\mathbf{u}} \leftarrow \mathbb{Z}_p^{nm}$ and $\mathbf{R} \leftarrow \mathbb{Z}_p^{k\times m}$ and output

$$\mathsf{sk}_f^* = \big(\mathbf{C}^\perp \cdot \mathsf{sk}_f^*[1] + \mathbf{a}^\perp \cdot \mathsf{sk}_f^*[2], [\mathbf{R}]_2\big) \quad \text{and} \quad f \tag{12}$$

  where

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

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

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

References

1. 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.
2. M. Abdalla, J. Gong, and H. Wee. Functional encryption for attribute-weighted sums from k-lin. In *CRYPTO*, 2020.
3. 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.
4. P. Ananth, A. Jain, H. Lin, C. Matt, and A. Sahai. Indistinguishability obfuscation without multilinear maps: New paradigms via low degree weak pseudorandomness and security amplification. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 284–332. Springer, Heidelberg, Aug. 2019.
5. 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.
6. D. Boneh, A. Sahai, and B. Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 573–592. Springer, Heidelberg, May / June 2006.
7. 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.
8. D. Boneh and B. Waters. A fully collusion resistant broadcast, trace, and revoke system. In A. Juels, R. N. Wright, and S. De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 211–220. ACM Press, Oct. / Nov. 2006.
9. 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.
10. 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.
11. S. Garg, A. Kumarasubramanian, A. Sahai, and B. Waters. Building efficient fully collusion-resilient traitor tracing and revocation schemes. In E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, editors, *ACM CCS 2010*, pages 121–130. ACM Press, Oct. 2010.
12. R. Gay. A new paradigm for public-key functional encryption for degree-2 polynomials. In A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 95–120. Springer, Heidelberg, May 2020.
13. R. Gay, A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from simple-to-state hard problems: New assumptions, new techniques, and simplification. Cryptology ePrint Archive, Report 2020/764, 2020.
14. J. Gong and H. Qian. Simple and efficient fe for quadratic functions. Cryptology ePrint Archive, Report 2020/1026, 2020.
15. 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.
16. R. Goyal, V. Koppula, and B. Waters. New approaches to traitor tracing with embedded identities. In D. Hofheinz and A. Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 149–179. Springer, Heidelberg, Dec. 2019.
17. 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.
18. 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.
19. A. Jain, H. Lin, and A. Sahai. Simplifying constructions and assumptions for io. *IACR Cryptology ePrint Archive*, 2019:1252, 2019.
20. A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from well-founded assumptions. Cryptology ePrint Archive, Report 2020/1003, 2020.
21. 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.
22. T. Ryffel, D. Pointcheval, F. Bach, E. Dufour-Sans, and R. Gay. Partially encrypted deep learning using functional encryption. In *Advances in Neural Information Processing Systems 32: NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 4519–4530, 2019.
23. 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 Concrete Scheme for Quadratic Functions

We present a self-contained description of our functional encryption scheme for quadratic functions specified by $\mathbf{f} \in \mathbb{Z}_p^{n_1 \times n_2}$ where

$$\mathbf{z}_1, \mathbf{z}_2 \mapsto (\mathbf{z}_1 \otimes \mathbf{z}_2)\mathbf{f}^\top$$

The scheme is SA-SIM-secure under the bilateral k-Lin assumption and the k'-Lin assumption in $\mathbb{G}_1, \mathbb{G}_2$. For the most efficient concrete instantiation (cf. Fig 1), we set $k = 2, k' = 1$.

– $\mathsf{Setup}(p, 1^{n_1}, 1^{n_2})$: Run $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(p)$. Sample

$$\mathbf{A}_1 \leftarrow \mathbb{Z}_p^{k \times n_1}, \mathbf{A}_2 \leftarrow \mathbb{Z}_p^{k' \times n_2}, \mathbf{A}_0 \leftarrow \mathbb{Z}_p^{k' \times (k'+1)}, \mathbf{W} \leftarrow \mathbb{Z}_p^{(k'+1) \times (k' n_1 + k n_2)},$$

and output

$$\mathsf{mpk} = \big(\mathbb{G}, [\mathbf{A}_0]_1, [\mathbf{A}_0\mathbf{W}]_1, [\mathbf{A}_1]_1, [\mathbf{A}_1]_2, [\mathbf{A}_2]_2\big) \quad \text{and} \quad \mathsf{msk} = \mathbf{W}$$

– $\mathsf{Enc}(\mathsf{mpk}, (\mathbf{z}_1, \mathbf{z}_2))$: Sample $\mathbf{s}_1 \leftarrow \mathbb{Z}_p^k, \mathbf{s}_0, \mathbf{s}_2 \leftarrow \mathbb{Z}_p^{k'}$ and output

$$\mathsf{ct} = \big([\underbrace{\mathbf{s}_1\mathbf{A}_1 + \mathbf{z}_1}_{\mathbf{y}_1}]_1, [\underbrace{\mathbf{s}_2\mathbf{A}_2 + \mathbf{z}_2}_{\mathbf{y}_2}]_2, [\underbrace{\mathbf{s}_0\mathbf{A}_0}_{\mathbf{c}_0}]_1, [\underbrace{\mathbf{s}_0\mathbf{A}_0\mathbf{W} + (\mathbf{s}_1 \otimes \mathbf{z}_2 \mid \mathbf{y}_1 \otimes \mathbf{s}_2)}_{\mathbf{y}_0}]_1\big) \in \mathbb{G}_1^{n_1} \times \mathbb{G}_2^{n_2} \times \mathbb{G}_1^{k'+1} \times \mathbb{G}_1^{k' n_1 + k n_2}$$

– $\mathsf{KeyGen}(\mathsf{msk}, \mathbf{f})$: Output

$$\mathsf{sk}_\mathbf{f} = \left[\mathbf{W} \cdot \begin{pmatrix} (\mathbf{A}_1 \otimes \mathbf{I}_{n_2})\mathbf{f}^\top \\ (\mathbf{I}_{n_1} \otimes \mathbf{A}_2)\mathbf{f}^\top \end{pmatrix}\right]_2 \in \mathbb{G}_2^{(k'+1) \times 1}$$

– $\mathsf{Dec}(\mathsf{sk}_\mathbf{f}, \mathbf{f}, \mathsf{ct})$: Parse $\mathsf{sk}_\mathbf{f} = [\mathbf{k}^\top]_2$ and output the discrete log of

$$[(\mathbf{y}_1 \otimes \mathbf{y}_2) \cdot \mathbf{f}^\top]_T \cdot e([\mathbf{c}_0]_1, [\mathbf{k}^\top]_2) \cdot e\left([\mathbf{y}_0]_1, \left[\begin{pmatrix} (\mathbf{A}_1 \otimes \mathbf{I}_{n_2})\mathbf{f}^\top \\ (\mathbf{I}_{n_1} \otimes \mathbf{A}_2)\mathbf{f}^\top \end{pmatrix}\right]_2\right)^{-1}$$

B Security Proof for Section 4

We complete the security proof for the scheme Π in Section 4.2.

Theorem 1. *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*

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

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.

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 parameters t and m. We prove Theorem 1 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

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

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.

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

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

with

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

where $(\mathbf{L}_{1,j},\mathbf{L}_{0,j}) \leftarrow \mathsf{lgen}(f_j)$, $\mathbf{T}_j \leftarrow \mathbb{Z}_p^{(k+1)\times t}$, $\mathbf{R}_j \leftarrow \mathbb{Z}_p^{k\times m}$, $\mathbf{c}$ is the randomness in ct^* and $\mathbf{C}^{\perp}$ is defined such that $(\mathbf{A}|\mathbf{c})^{\top}(\mathbf{C}^{\perp}|\mathbf{a}^{\perp}) = \mathbf{I}_{k+1}$ (cf. Setup* in Section 4.2). By basic linear algebra, we have Game$_1$ = Game$_2$.

Game$_3$: Identical to Game$_2$ except that we replace Setup, Enc with Setup*, Enc* where ct* is given by

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

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

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

where $\tilde{\mathbf{w}}, \tilde{\mathbf{v}}$ are given in msk* (output by Setup*) and $\tilde{\mathbf{u}} \leftarrow \mathbb{Z}_p^{kn}, \mathbf{t}_j \leftarrow \mathbb{Z}_p^{t}, \mathbf{R}_j \leftarrow \mathbb{Z}_p^{k\times m}$. We claim that Game$_2 \approx_s$ Game$_3$. This follows from the following statement: for any full-rank $(\mathbf{A}|\mathbf{c})$, we have

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

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

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

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

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

where Q is the number of key queries.

Game$_5$: Identical to Game$_4$ except that we replace KeyGen_4^* with KeyGen^*; this is the ideal game. We claim that Game$_4 \approx_s$ Game$_5$. This follows from the privacy of partial garbling scheme in Section 4.1.

We use $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{xx}}(\lambda)$ to denote the advantage of adversary $\mathcal{A}$ in Game$_{\mathsf{xx}}$. We prove the following lemmas showing the indistinguishability of adjacent games listed above.

Lemma 2 (Game$_0 \approx_c$ Game$_1$)**.** *For all* $\mathcal{A}$*, there exists* $\mathcal{B}_1$ *with* $\mathsf{Time}(\mathcal{B}_1) \approx \mathsf{Time}(\mathcal{A})$ *such that*

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

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

The proof is the same as before, except we replace $\boxed{\mathbf{cW}}$, $\mathbf{z}^* - \tilde{\mathbf{w}}$ in $\mathsf{sk}_{f_j}[2]$ with $\boxed{\mathbf{cWM}_j}$, $\mathbf{z}^*\mathbf{M}_j - \tilde{\mathbf{w}}\mathbf{M}_j$

Proof (of Lemma 3). Recall that the difference between the two games lies in ct^* and $\mathsf{sk}_{f_j}[2]$: instead of computing

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

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

in Game_2, we compute

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

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

in Game_3.

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

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

where $\mathbf{U}, \mathbf{W}, \mathbf{V}, \tilde{\mathbf{w}}, \tilde{\mathbf{v}}$ are sampled as in Setup^* and $\tilde{\mathbf{u}} \leftarrow \mathbb{Z}_p^{kn}, \mathbf{T}_j \leftarrow \mathbb{Z}_p^{(k+1)\times t}, \mathbf{t}_j \leftarrow \mathbb{Z}_p^t$. 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 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 4 ($\mathsf{Game}_3 \approx_c \mathsf{Game}_4$)**.** *For all $\mathcal{A}$, there exists $\mathcal{B}_2$ with $\mathsf{Time}(\mathcal{B}_2) \approx \mathsf{Time}(\mathcal{A})$ such that*

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

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.

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

The proof is the same as before except we replace $\mathbf{z}^*$ in $\mathsf{sk}_{f_j}[2], \mathsf{pgb}, \mathsf{pgb}^*$ with $\mathbf{z}^*\mathbf{M}_j$ and $\tilde{\mathbf{w}}$ in $\mathsf{sk}_{f_j}[2]$ with $\tilde{\mathbf{w}}\mathbf{M}_j$.

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

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

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

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

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:

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

where

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

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