Dynamic Decentralized Functional Encryptions from Pairings in the Standard Model

Duy Nguyen[0009−0002−7892−9146]

Telecom Paris, Institut Polytechnique de Paris, France
dinh.nguyen@telecom-paris.fr

Abstract. Dynamic Decentralized Functional Encryption (DDFE), introduced by Chotard et al. (CRYPTO'20), stands as a robust generalization of (Multi-Client) Functional Encryption. It enables users to dynamically join and contribute private inputs to individually-controlled joint functions, all without requiring a trusted authority. Agrawal et al. (TCC'21) further extended this line of research by presenting the first DDFE construction for function-hiding inner products (FH-IP-DDFE) in the random oracle model (ROM).
Recently, Shi et al. (PKC'23) proposed the first Multi-Client Functional Encryption construction for function-hiding inner products based on standard assumptions without using random oracles. However, their construction still necessitates a trusted authority, leaving the question of whether a fully-fledged FH-IP-DDFE can exist in the standard model as an exciting open problem.
In this work, we provide an affirmative answer to this question by proposing a FH-IP-DDFE construction based on the Symmetric External Diffie-Hellman (SXDH) assumption in the standard model. Our approach relies on a novel zero-sharing scheme termed as Updatable Pseudorandom Zero Sharing, which introduces new properties related to updatability in both definition and security models. We further instantiate this scheme in groups where the Decisional Diffie-Hellman (DDH) assumption holds.
Moreover, our proposed pseudorandom zero sharing scheme serves as a versatile tool to enhance the security of pairing-based DDFE constructions for functionalities beyond inner products. As a concrete example, we present the first DDFE for attribute-weighted sums in the standard model, complementing the recent ROM-based construction by Agrawal et al. (CRYPTO'23).

Keywords: dynamic, decentralized, functional encryption, pairing, standard model

1 Introduction

1.1 Dynamic Decentralized Functional Encryption

Functional Encryption. Functional Encryption (FE) is a robust cryptographic paradigm introduced by Sahai and Waters [SW05, BSW11]. It addresses the all-or-nothing limitations of standard public-key cryptosystems by offering fine-grained control over access to encrypted data through functional decryption keys. More precisely, each ciphertext ct_x encrypts a specific value x, and each decryption key dk_f encapsulates a function f. When a receiver uses dk_f to decrypt ct_x, what it can learn is only the result $f(x)$, and nothing more about x. Depending on a specific choice of functionality class, FE encompasses various advanced encryption schemes, including Identity-Based Encryption [Sha84, BF01], Attribute-Based Encryption [SW05, GPSW06], Predicate Encryption [BW07, KSW08], and more.

Since its introduction, Functional Encryption (FE) has emerged as a dynamic area of research. One direction in this field involves the construction of FE schemes for general functionalities and the exploration of their relationships with other cryptographic notions. A variety of elegant works have been published in this regard [AJ15, BV15]. However, it is worth noting that all known FE constructions for general functionalities rely on non-standard cryptographic assumptions, such as indistinguishability obfuscation, single-input FE for circuits, or multilinear maps. Consequently, this reliance poses challenges in practical implementations.

On another front, Abdallah et al. [ABDP15] addressed the challenge of constructing FE based on standard assumptions for restricted, yet expressive, classes of functions. In their work, they presented

FE constructions for inner products[1] (IP-FE), where each private input is represented as a vector $\boldsymbol{x}$, each function is represented by a vector $\boldsymbol{y}$, and decryption yields $\boldsymbol{x}^\top \cdot \boldsymbol{y}$.

Within the realm of practical FE based on standard assumptions for computation over encrypted data, significant improvements have been made in terms of security [BJK15, ALS16, ALMT20], functionality [BCFG17, AGW20, ACGU20], efficiency [CLT18, TT18, DP19, MKMS22] and support for multi-user scenarios [ACF+18, CDG+18, CDSG+20, ZLZ+24].

Functional Encryption for Multiple Users. The concepts of Multi-Input Functional Encryption (MIFE) and Multi-Client Functional Encryption (MCFE) were introduced in [GGG+14, GKL+13]. These concepts generalize FE to enable multiple clients to independently contribute their encrypted individual inputs to the computation of joint functions. Concretely, with the help of possibly a trusted authority, each slot owner S_i receives a private encryption key sk_i and encrypts an input x_i under some label ℓ as $\mathsf{ct}_{x_i,\ell}$. By collecting ciphertexts of all slot owners $(\mathsf{ct}_{x_i,\ell})_i$ under a same label and a functional decryption key dk_f generated from the authority's master secret key msk, a receiver can obtain no more information than $f((x_i)_i)$.

In the multi-client setting, each slot owner is assumed to be an independent client: the security guarantees that only ciphertexts under a same label can be decrypted together, and the input privacy of honest clients holds even when a subset of clients is corrupted. In the multi-input setting, all labels are assumed to be the same, all slot owners are originally assumed to be the same user (yet each input can be sent independently instead of simultaneously as in single-input FE), and then no corruption is considered.

Therefore, any MCFE designed for a specific functionality automatically implies a MIFE for the same functionality by using a fixed label for all encryptions. Conversely, an MIFE designed for general functions directly implies an MCFE for general functions, as the label can be contained in every plaintext, and the function can verify that every slot uses the same label. Considering practical FE schemes in the literature, this equivalence doesn't hold for restricted classes of functions such as inner-products.

Since their introduction, the research line on practical MIFE/MCFE constructions has witnessed a dynamic array of works seeking developments in the aspects of security [CDG+18, LT19, AGT22, SV23], functionality [CDSG+20, ACGU20, AGT21a, NPP22, ATY23], and in the relation with single-input FE [ABG19].

Decentralized Multi-Client Functional Encryption. Standard MCFE models often encounter the key escrow problem, stemming from the reliance on a trusted authority. To address this challenge, Chotard *et al.* [CDG+18] introduced the concept of decentralized MCFE (DMCFE). In DMCFE, the need for a trusted authority is entirely removed, granting each client complete control over their encrypted data and the generation of functional decryption keys.

In this paradigm, each client is require to join together a one-time interactive setup to acquire their individual secret key sk_i. With this key, each client can generate ciphertexts similar to those in standard MCFE and decryption key shares $\mathsf{dk}_{f,i}$ for a function f agreed upon by all clients. To decrypt, a receiver must collect decryption key shares from all clients; otherwise, decryption yields no information. This decentralized approach to key generation eliminates the necessity for a central authority, thereby enhancing the security model of MCFE.

The development of DMCFE since the introduction of the first scheme for inner products [CDG+18] has seen significant progress on various aspects. These advancements include expanding functionality, as seen in constructions for attribute-weighted sums [ATY23]. Efforts to enhance security have led to DMCFE constructions for function-hiding inner products [AGT21b], for lattice-based assumptions in the standard model [LT19], and for optimal security notions [NPP23b]. The relations between DMCFE and other FE notions have been established in these works [ABG19, ABKW19]. Practical features like verifiability [NPP23a] and robustness [LWG+23] have also been integrated, enhancing the applicability of DMCFE in various real-world scenarios.

Dynamic Decentralized Functional Encryption. Dynamic Decentralized Functional Encryption (DDFE), as introduced by Chotard et al. [CDSG+20], extends the concept of DMCFE by preserving all decentralized features while enabling the join of new clients at various stages during the lifetime of the system through a non-interactive setup.

[1] In the literature, there are other synonyms for inner products such as linear functions or weighted sums.

In DDFE, each client in a universe $\mathcal{PK}$ can independently generate a pair of public key pk and private key $\mathsf{sk}_{\mathsf{pk}}$. The public key pk will be published on the cloud and updated by other clients in the system. Compared to DMCFE, the decryption algorithm in DDFE is additionally more flexible, as it allows a combination between ciphertexts $(\mathsf{ct}_{m_{\mathsf{pk}}})_{\mathsf{pk}\in\mathcal{U}_M}$ of clients in any list $\mathcal{U}_M \subset \mathcal{PK}$, and decryption keys $(\mathsf{dk}_{k_{\mathsf{pk}}})_{\mathsf{pk}\in\mathcal{U}_K}$ of clients in any list $\mathcal{U}_K \subset \mathcal{PK}$.

The first DDFE schemes [CDSG+20] were constructed to support functionalities such as sums, inner products, and all-or-nothing message encapsulations. As subsequent works, DDFE constructions for larger classes of functions such as function-hiding inner products and attributed-weighted sums have been introduced in [AGT21b, ATY23] respectively.

Multi-Party Functional Encryption. In addition to the above notions of functional encryption, there are other concepts designed for various multi-user settings. These include those supporting distributed keys, such as Decentralized Attribute Based Encryption with Policy Hiding [MJ18] and Multi-Authority Functional Encryption [Cha07, LW11, BCFG17], as well as those supporting both distributed ciphertexts and keys, such as Ad Hoc MIFE [ACF+20]. All these notions fall under the umbrella of Multi-Party Functional Encryption [AGT21b].

However, in this paper, our focus will be exclusively on DDFE, which stands out as the most generalized notion within Multi-Party Functional Encryption and can support both distributed ciphertexts and keys.

An Open Problem in DDFE. To the best of our knowledge, practical constructions for DDFE based on standard assumptions remain limited to the classes of function-hiding inner products (FH-IP) and function-revealing attribute-weighted sums (AWS), which strictly capture the class of function-revealing inner products (IP). These functionalities can be described more precisely as follows:

- In DDFE for FH-IP: each client, identified by a public key $\mathsf{pk} \in \mathcal{PK}$, uses its own corresponding secret key $\mathsf{sk}_{\mathsf{pk}}$ to encrypt its private input x_{pk} as $\mathsf{ct}_{\boldsymbol{x}_{\mathsf{pk}},\ell_M,\mathcal{U}_M}$ under a public message label ℓ_M and for a public user list $\mathcal{U}_M$. Similarly, each client uses $\mathsf{sk}_{\mathsf{pk}}$ to generate a decryption key for its private vector $\boldsymbol{y}_{\mathsf{pk}}$ as $\mathsf{dk}_{\boldsymbol{y}_{\mathsf{pk}},\ell_K,\mathcal{U}_K}$ under a public key label ℓ_K and for a public user list $\mathcal{U}_K$. The labels ℓ_M and ℓ_K impose constraints on which messages and functions can be aggregated together, respectively, while the lists $\mathcal{U}_M$ and $\mathcal{U}_K$ specifies the sets of parties whose ciphertexts and decryption keys can be combined, respectively. During decryption, the FH-IP functionality verifies the conditions $[\mathcal{U}_M = \mathcal{U}_K]$, $[(\ell_M,\mathcal{U}_M)$ is consistent across all ciphertexts] and $[(\ell_K,\mathcal{U}_K)$ is consistent across all decryption keys]. If these conditions are met, the FH-IP functionality outputs
$$\sum_{\mathsf{pk}\in\mathcal{U}_K} \boldsymbol{x}_{\mathsf{pk}}^\top \cdot \boldsymbol{y}_{\mathsf{pk}};$$
otherwise, it outputs nothing. The function-hiding security ensures that no additional information on individual $\boldsymbol{x}_{\mathsf{pk}}$ and $\boldsymbol{y}_{\mathsf{pk}}$ is revealed.
- In DDFE for AWS: each client pk chooses a user list $\mathcal{U}_M$, a label ℓ to encrypt its AWS inputs $\{\boldsymbol{z}_{\mathsf{pk},j}\}_{j\in[N_{\mathsf{pk}}]}$ which are private and $\{\boldsymbol{x}_{\mathsf{pk},j}\}_{j\in[N_{\mathsf{pk}}]}$ which are public. For key generation, each client pk chooses a set of users $\mathcal{U}_K$ and a list of arithmetic branching programs (ABPs) $\boldsymbol{f} = \{f_{\mathsf{pk}}\}_{\mathsf{pk}\in\mathcal{U}_K}$. During decryption, the AWS functionality verifies the conditions $[\mathcal{U}_M = \mathcal{U}_K]$, $[(\ell,\mathcal{U}_M)$ is consistent across all ciphertexts] and [$\boldsymbol{f}$ is consistent across all decryption keys]. If these conditions are met, the AWS functionality outputs
$$\sum_{\mathsf{pk}\in\mathcal{U}_K}\sum_{j\in[N_{\mathsf{pk}}]} f_{\mathsf{pk}}(\boldsymbol{x}_{\mathsf{pk},j})^\top \cdot \boldsymbol{z}_{\mathsf{pk},j};$$
otherwise, it outputs nothing. The function-revealing security ensures that no additional information on individual $\{\boldsymbol{z}_{\mathsf{pk},j}\}_{j\in[N_{\mathsf{pk}}]}$ is revealed.

The constructions for function-revealing inner products, function-hiding inner products, and attribute-weighted sums are provided in [CDSG+20, AGT21b, ATY23] respectively, representing the current state of the art in their respective areas[2]. However, it's important to highlight that these constructions rely on random oracles for their security proofs.

[2] An advantage of the construction for function-revealing inner products is that it can be instantiated in a pairing-free group, while all FE constructions for function-hiding inner products relies on pairing groups.

On the other hand, in the realm of (Decentralized) MCFE for (function-hiding) inner products, there exists a series of elegant works [LT19, ABG19, SV23] aimed at enhancing security by eliminating the use of random oracles and ensuring the security of the respective (Decentralized) MCFE in the standard model. Therefore, our objective is to contribute to this line of research by addressing the following question:

> *Do there exist any DDFE constructions for function-hiding inner products and for attribute-weighted sums that are secure under standard assumptions in the standard model?*

This work provides an affirmative answer to this question for both function classes.

1.2 Our Contributions

The contributions can be listed as follows:

- **Novel Pseudorandom Zero Sharing**: We introduce a new concept called *Updatable Pseudorandom Zero Sharing*. Compared to standard pseudorandom zero sharing, this notion additionally offers the following key properties:
  1. it enables the local update from a zero share into a new one;
  2. its security guarantees the pseudorandomness for the updated shares even if their corresponding non-updated shares are revealed; this security holds against an unbounded subset of corrupted parties;
  3. its updating algorithm can support the *bilinear update* property, which makes the scheme friendly with pairing-based constructions;

  In addition to the definition and security models, we also provide a concrete instantiation of this notion in Decisional Diffie-Hellman Assumption (DDH) groups.
- **Dynamic Decentralized Functional Encryptions Without RO**: Using the Updatable Pseudorandom Zero Sharing scheme in DDH groups as a building block, we provide the first DDFE constructions for function-hiding inner products and attribute-weighted sums that are secure in the standard model. The security of these constructions relies on the SXDH assumption in the selective-symmetric setting. For function-hiding inner products, an incidental trade-off[3] is that the ciphertext size per client increases to $O(d+n)$ from $O(d)$, where d is an inner-product dimension, and n is the size of a user list $\mathcal{U}$ whose ciphertexts and decryption keys can be combined. Further comparative details are provided in Figure 1.

1.3 Technical Overview

Random Oracles in Prior Schemes. In the function-hiding (decentralized) multi-client setting, one of the fundamental security requirements is that only inputs encrypted under a same message label and only keys generated under a same key label can be decrypted together, otherwise the decryption should output nothing.

Thus, a common strategy in constructing (decentralized) MCFE schemes, including those for (function-hiding) inner products and attribute-weighted sums, is to rely on a (correlated) one-time-pad technique: the one-time pads should be freshly generated by each pair of message and key labels, as well as by each client, to independently randomize the information corresponding to each client's private pair of input and key object. This strategy is also employed in the constructions of DDFE, as we will describe below.

Without obscuring the reliance on random oracles for its security, we can simplify the DDFE scheme for FH-IP in [AGT21b] as follows: leveraging a private use of function-hiding IPFE, each client $\mathsf{pk} \in \mathcal{U}$ encrypts a private $\boldsymbol{x}_{\mathsf{pk}}$ as

$$\mathsf{ct}_{\mathsf{pk}} = \mathsf{IPE.Enc}(\mathsf{sk}_{\mathsf{pk}}, [\boldsymbol{x}_{\mathsf{pk}}, \mathbf{0}, h_{\ell_M,\mathcal{U}}, 0]_1)$$

[3] This trade-off and the selective-symmetric security does not compromise the solution's completeness regarding the open question posed in [SV23], which queries whether a DDFE for function-hiding inner products can exist from standard assumptions without relying on random oracles.

Scheme	Function Class	Function Hiding	(Dynamic) Decentralized	Without ROM	Assumptions	Per-client CT size
[CDG+18]	IP	✗	✓	✗	SXDH	$O_\lambda(d)$
[ABG19]	IP	✗	✓	✓	IPFE	$O_\lambda(d \cdot n)$
[LT19]	IP	✗	✓	✓	LWE	$O_\lambda(d)$
[CDSG+20]	IP	✗	✓	✗	IPFE + DDH	$O_\lambda(d)$
[ABM+20]	IP	✗	✓	✗	DCR	$O_\lambda(d)$
[AGT21b]	IP	✓	✓	✗	SXDH +FH-IPFE	$O_\lambda(d)$
[SV23]	IP	✓	✗	✓	DLin + FH-IPFE	$O_\lambda(d)$
Our FH-IP DDFE	IP	✓	✓	✓	SXDH +FH-IPFE	$O_\lambda(d+n)$
[ATY23]	AWS	✗	✓	✗	SXDH +AWSw/IP-FE	$O_\lambda(N+k)$
Our AWS DDFE	AWS	✗	✓	✓	SXDH +AWSw/IP-FE	$O_\lambda(N+n)$

Fig. 1. Comparison with prior (Decentralized) MCFE schemes. The notation $O_\lambda(\cdot)$ indicates that terms related to the security parameter λ are hidden. We let d be an inner-product dimension, N be an $\mathsf{poly}(\lambda)$-unbounded number of AWS inputs, n be an number of clients whose ciphertexts and decryption keys can be combined, and k be the parameter of the MDDH assumption. For DDFE, all constructions achieve sel-sym-IND security.

where $[h_{\ell_M,\mathcal{U}}]_1 = \mathcal{H}(\ell_M,\mathcal{U})$ and $\mathcal{H}$ is a hash function onto the group modeled as a random oracle. The decryption key for a private $\boldsymbol{y}_{\mathsf{pk}}$ is generated as

$$\mathsf{dk}_{\mathsf{pk}} = \mathsf{IPE.DKGen}(\mathsf{sk}_{\mathsf{pk}}, [\boldsymbol{y}_{\mathsf{pk}}, \mathbf{0}, z_{\mathsf{pk},\ell_K}, 0]_2)$$

where z_{pk,ℓ_K} is generated on label ℓ_K by a pseudorandom zero-sharing (PZS) scheme between parties in $\mathcal{U}$ so that $\sum_{\mathsf{pk}\in\mathcal{U}} z_{\mathsf{pk},\ell_K} = 0$. On one hand, the correctness holds as $\sum_{\mathsf{pk}\in\mathcal{U}} z_{\mathsf{pk},\ell_K} \cdot h_{\ell_M,\mathcal{U}} = 0$ in the sum of all FH-IPFE decryptions.

On the other hand, via zero slots, and by using hybrids relying on the function-hiding security of each client's FH-IPFE in the non-corrupted set $\mathcal{H}$ and each key label ℓ_K, one can move the term $[z_{\mathsf{pk},\ell_K}]_2$ from $\mathsf{dk}_{\mathsf{pk}}$ to the term $[z_{\mathsf{pk},\ell_K} \cdot h_{\ell_M,\mathcal{U}}]_1$ in $\mathsf{ct}_{\mathsf{pk}}$. At this step, one will have

$$[z_{\mathsf{pk},\ell_K} \cdot h_{\ell_M,\mathcal{U}}]_1 \overset{\mathsf{PZS},\mathsf{SXDH}}{\approx} R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K}$$

where $R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K}$ are generated uniformly random for the relation $\sum_{\mathsf{pk}\in\mathcal{H}} R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K} = -\sum_{\mathsf{pk}\in\mathcal{U}\setminus\mathcal{H}} z_{\mathsf{pk},\ell_K} \cdot h_{\ell_M,\mathcal{U}}$. By the admissibility condition, each $R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K}$ serves as a correlated one-time pad that randomizes each term $\boldsymbol{x}^{b\top}_{\mathsf{pk}} \cdot \boldsymbol{y}^{b}_{\mathsf{pk}}$ in the same way it randomizes $\boldsymbol{x}^{0\top}_{\mathsf{pk}} \cdot \boldsymbol{y}^{0}_{\mathsf{pk}}$, where b is the challenge bit of the security game. It is important to note that for the SXDH assumption to hold, $[h_{\ell_M,\mathcal{U}}]_1$ must be a uniformly random group element to all clients, which requires the use of a random oracle.

A more recent FH-IP construction in [SV23] provides another helpful one-time-pad technique to avoid the reliance on random oracles for the non-decentralized multi-client setting. The construction can be briefly described as follows: leveraging a private use of function-hiding IPFE, each client $i \in [n]$ encrypts a private $\boldsymbol{x}_i$ as

$$\mathsf{ct}_i = \mathsf{IPE.Enc}(\mathsf{sk}_i, [\boldsymbol{x}_i, \mathbf{0}, z_{i,\ell_M} + a_i \cdot \mu_{i,\ell_M}, \mu_{i,\ell_M}, 0])$$

where z_{i,ℓ_M} is generated on label ℓ_M by a pseudorandom zero-sharing, the value a_i is a random key shared between client i and the trusted authority, and μ_{i,ℓ_M} is a fresh randomness. The decryption key for a private $\boldsymbol{y}_i$ is generated as

$$\mathsf{dk}_i = \mathsf{IPE.DKGen}(\mathsf{sk}_i, [\boldsymbol{y}_i, \mathbf{0}, \rho_{\ell_K}, -\rho_{\ell_K} a_i, 0])$$

where ρ_{ℓ_K} is the same random value in dk_i for all $i \in [n]$. The correctness holds as $\sum_{i\in[n]} z_{i,\ell_M} \cdot \rho_{\ell_K} = 0$ in the sum of all FH-IPFE decryptions, and the MCFE security holds under the Decisional Linear

assumption in the standard model. Unfortunately, extending this construction to the decentralized setting appears challenging, as it crucially requires a trusted authority to generate the same random ρ_{ℓ_K} for all FH-IPFE decryption keys.

It is also important to note that the generic transformation from single-input FE to decentralized MCFE for function-revealing inner products in [ABG19] cannot be extended to the function-hiding setting, as the transformation necessitates each client to know the entire joint inner-product function.

Updatable Pseudorandom Zero-Sharing. In Section 3, we provide a definition and security models for the new notion of Updatable Pseudorandom Zero-Sharing (UZS). The formalization will be useful for the black-box use in DDFE schemes and any potential improvements.

We provide a brief overview of our DDH-based construction in Figure 2, which will serve as a new one-time-pad technique leading to the security in the standard model for pairing-based DDFE constructions. When compared to the pseudorandom zero-sharing schemes in earlier works [BIK+17, ABG19], it's notable that zero shares in our scheme are DDH group elements that sum up to the group identity, instead of scalars.

Algorithm	Return
$\mathsf{SetUp}(\lambda)$	a group $\mathbb{G}$ and pseudorandom functions $(\mathsf{PRF}, \mathsf{PRF}')$.
$\mathsf{KeyGen}()$	$\mathsf{sk}_i = (k_{i,j} = k_{j,i})_{j\in[n]\setminus[i]}$ for each party P_i.
$\mathsf{SeedGen}(\mathsf{sk}_i, \ell_M)$	$\mathsf{seed}_{i,\ell_M} = ([(-1)^{i<j}\mathsf{PRF}_{k_{i,j}}(\ell_M)])_{j\in[n]}$.
$\mathsf{TokGen}(\mathsf{sk}_i, \ell_K)$	$\mathsf{token}_{i,\ell_K} = (\mathsf{PRF}'_{k_{i,j}}(\ell_K))_{j\in[n]}$.
$\mathsf{SeedUpt}(\mathsf{seed}_{i,\ell_M}, \mathsf{token}_{i,\ell_K})$	$\mathsf{seed}_{i,\ell_M\|\|\ell_K} = ([(-1)^{i<j}\mathsf{PRF}_{k_{i,j}}(\ell_M)\cdot\mathsf{PRF}'_{k_{i,j}}(\ell_K)])_{j\in[n]}$.
$\mathsf{ShareEval}(\mathsf{seed}_{i,\ell})$	$\mathsf{share}_{i,\ell} = \sum_{j\in[n]\setminus i}[c_{i,j,\ell}]$ where $\mathsf{seed}_{i,\ell} = ([c_{i,j,\ell}])_{j\in[n]\setminus i}$.

Fig. 2. Construction for UZS in DDH groups between n parties

We demonstrate how the scheme achieves correctness and security using matrix representation:

- The matrix $[\boldsymbol{A}_{\ell_M}]$ (see Figure 3) serves as a seeding matrix, with each i-th row representing seed_{i,ℓ_M}. This seeding matrix adopts an anti-symmetric form: $A_{\ell_M,(i,i)} = 0$ for all $i \in [n]$ and $A_{\ell_M,(i,j)} = -A_{\ell_M,(j,i)}$ for all $(i, j \in [n], i \neq j)$.

$$[\boldsymbol{A}_{\ell_M}] = \begin{bmatrix} 0 & -\mathsf{PRF}_{k_{1,2}}(\ell_M) & \cdots & -\mathsf{PRF}_{k_{1,n}}(\ell_M) \\ \mathsf{PRF}_{k_{2,1}}(\ell_M) & 0 & \cdots & -\mathsf{PRF}_{k_{2,n}}(\ell_M) \\ \vdots & \vdots & \ddots & \vdots \\ \mathsf{PRF}_{k_{n,1}}(\ell_M) & \mathsf{PRF}_{k_{n,2}}(\ell_M) & \cdots & 0 \end{bmatrix} \begin{matrix} \\ \rightarrow \mathsf{seed}_{1,\ell_M} = ([A_{\ell_M,(1,j)}])_{j\in[n]\setminus\{1\}} \\ \rightarrow \mathsf{seed}_{2,\ell_M} = ([A_{\ell_M,(2,j)}])_{j\in[n]\setminus\{2\}} \\ \\ \rightarrow \mathsf{seed}_{n,\ell_M} = ([A_{\ell_M,(n,j)}])_{j\in[n]\setminus\{n\}} \end{matrix}$$

Fig. 3. Seeding matrix $[\boldsymbol{A}_{\ell_M}] \in \mathbb{G}^{n\times n}$.

- The matrix $\boldsymbol{B}_{\ell_K}$ (see Figure 4) serves as an updating matrix, with each i-th row representing $\mathsf{token}_{i,\ell_K}$. This updating matrix adopts a symmetric form: $B_{\ell_K,(i,i)} = 0$ for all $i \in [n]$ and $B_{\ell_K,(i,j)} = B_{\ell_K,(j,i)}$ for all $(i, j \in [n], i \neq j)$.
- The matrix $[\boldsymbol{C}_{\ell_M||\ell_K}]$ (see Figure 5) serves as an updated seeding matrix, with each i-th row representing $\mathsf{seed}_{i,\ell_M||\ell_K}$. This updated matrix results from a Hadamard product between $[\boldsymbol{A}_{\ell_M}]$ and $\boldsymbol{B}_{\ell_K}$, which preserves the anti-symmetric form from $[\boldsymbol{A}_{\ell_M}]$.

In both cases, when $\ell = \ell_M$ or $\ell = \ell_M||\ell_K$, each $\mathsf{share}_{i,\ell}$ computes the sum of all entries on the i-th row of an antisymmetric matrix. Therefore, $\sum_{i\in[n]}\mathsf{share}_{i,\ell}$ computes the sum of all entries in the matrix, resulting in the group identity $[0]$. This implies the correctness of the scheme.

At a high level, the standard security of a pseudorandom zero-sharing scheme guarantees that when the adversary corrupts a subset of parties (up to $n-2$ out of n) and computes their shares

$$\boldsymbol{B}_{\ell_K} = \begin{pmatrix} 0 & \mathsf{PRF}'_{k_{1,2}}(\ell_K) & \cdots & \mathsf{PRF}'_{k_{1,n}}(\ell_K) \\ \mathsf{PRF}'_{k_{2,1}}(\ell_K) & 0 & \cdots & \mathsf{PRF}'_{k_{2,n}}(\ell_K) \\ \vdots & \vdots & \ddots & \vdots \\ \mathsf{PRF}'_{k_{n,1}}(\ell_K) & \mathsf{PRF}'_{k_{n,2}}(\ell_K) & \cdots & 0 \end{pmatrix} \begin{matrix} \\ \longrightarrow \mathsf{token}_{1,\ell_K} = (B_{\ell_K,(1,j)})_{j\in[n]} \\ \longrightarrow \mathsf{token}_{2,\ell_K} = (B_{\ell_K,(2,j)})_{j\in[n]} \\ \\ \longrightarrow \mathsf{token}_{n,\ell_K} = (B_{\ell_K,(n,j)})_{j\in[n]} \end{matrix}$$

Fig. 4. Updating matrix $\boldsymbol{B}_{\ell_M} \in \mathbb{Z}_p^{n\times n}$.

on its own, the shares of the remaining honest users are computationally indistinguishable from the (correlated) random distribution. Our scheme satisfies this standard security while additionally providing indistinguishability from a random distributtion for the updated shares: for a fixed set of corrupted users, even when the adversary has access to the seeding matrix $[\boldsymbol{A}_{\ell_M}]$ and thereby has access to all the shares of the honest users on ℓ_M, indistinguishability still holds for the updated honest users' shares in $[\boldsymbol{C}_{\ell_M||\ell_K}]$. Notably, the Hadamard product maintains the integrity of the honest entries from $[\boldsymbol{A}_{\ell_M}]$ and $\mathcal{B}_{\ell_K}$ to $[\boldsymbol{C}_{\ell_M||\ell_K}]$.

By the DDH assumption, we have $[\mathsf{PRF}_{k_{i,j}}(\ell_M)\mathsf{PRF}'_{k_{i,j}}(\ell_K)] \overset{\mathsf{DDH}}{\approx} \mathsf{RF}_{(i,j)}(\ell_M||\ell_K)$ for each honest pair (i,j) (see Figure 5), which implies that the honest shares in $[\boldsymbol{C}_{\ell_M||\ell_K}]$ are indepedently random from those in $[\boldsymbol{A}_{\ell_M}]$. Moreover, by using the Multi-DDH assumption, which tightly reduces to the DDH assumption using the random-self reducibility, the indistinguisability of the honest updated shares on $(\ell_M||\ell_K)$ holds for a polynomial number of labels ℓ_M given a label ℓ_K.

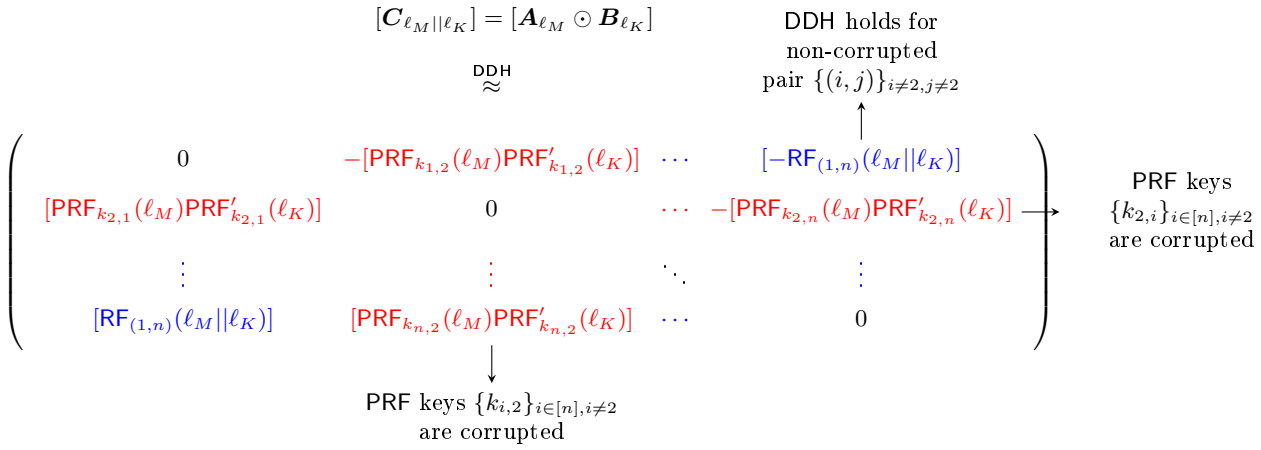

Fig. 5. Indistinguishabilities for entries in the updated seeding matrix $[\boldsymbol{C}_{\ell_M||\ell_K}]$, where RF denotes a random function. For simplicity, we assume that only user P_2 is corrupted by the adversary.

A concrete construction for the dynamic setting is provided in Section 3.3, which relies on a non-interactive key exchange protocol NIKE, a pseudorandom function PRF and the DDH assumption. Its security for a restricted setting of *one-time-update* and static corruption is provided in Theorem 1.

From UZS to FH-IP-DDFE Without RO. For each client $\mathsf{pk} \in \mathcal{U}$, the encryption of $\boldsymbol{x}_{\mathsf{pk}}$ under $(\mathcal{U}, \ell_M)$ and the generation of decryption key for $\boldsymbol{y}_{\mathsf{pk}}$ under $(\mathcal{U}, \ell_K)$ can be briefly described as follows:

$$\mathsf{ct}_{\mathsf{pk}} = \mathsf{IPE.Enc}_{(1^{2d+|\mathcal{U}|},\mathsf{sk}_{\mathsf{pk}})}([\boldsymbol{x}_{\mathsf{pk}}, \mathbf{0}^d, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}, 0]_1);$$
$$\mathsf{dk}_{\mathsf{pk}} = \mathsf{IPE.DKGen}_{(1^{2d+|\mathcal{U}|},\mathsf{sk}_{\mathsf{pk}})}([\boldsymbol{y}_{\mathsf{pk}}, \mathbf{0}^d, \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K}, 0]_2)$$

Here, d is an inner-product dimension, $[\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}]_1 = \mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell}$ and $\boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K} = \mathsf{token}_{\mathsf{pk},\mathcal{U},\ell_K}$ are generated using the UZS scheme. We note that for each user list $\mathcal{U}$, every client $\mathsf{pk} \in \mathcal{U}$ uses $\mathsf{sk}_{\mathsf{pk}}$ to pseudorandomly initialize a function-hiding IPE for inner products of length $(2d + |\mathcal{U}|)$. This is determined by the size $(|\mathcal{U}| - 1)$ of the UZS seeds and tokens. The correctness of the scheme holds

since

$$\begin{aligned}\mathsf{IPE.Dec}_{(1^{2d+|\mathcal{U}|},\mathsf{sk}_{\mathsf{pk}})}(\mathsf{ct}_{\mathsf{pk}},\mathsf{dk}_{\mathsf{pk}}) &= \boldsymbol{x}_{\mathsf{pk}}^{\top}\cdot\boldsymbol{y}_{\mathsf{pk}}+\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}^{\top}\cdot\boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K}\\ &= \boldsymbol{x}_{\mathsf{pk}}^{\top}\cdot\boldsymbol{y}_{\mathsf{pk}}+\mathsf{share}_{\mathsf{pk},\mathcal{U},\ell_M||\ell_K}\end{aligned}$$

by the bilinear-update property of the UZS scheme.

The security of the scheme holds in the symmetric and selective setting, under the one-key-per-label restriction. To obtain this indistinguishability, we use a sequence of hybrid games (see Figure 6): by each key label ℓ_K, we replace $\boldsymbol{y}^b$ with $\boldsymbol{y}^0$ in every honest decryption key under ℓ_K. Eventually, we change $\boldsymbol{x}^b$ to $\boldsymbol{x}^0$ in every honest encryption under all message labels to completely eliminate the challenge bit b. This strategy is closely similar to those used in the earlier works [AGT21b, SV23], with the exception of where the security of UZS applies in $\mathbf{G}^{\star}_{\ell_K.2}$ and $\mathbf{G}^{\star}_{\ell_K.4}$ in Figure 7.

During these steps, we rely on update indistinguishability to replace each honest updated share $[\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}]_1$ with a correlated random value $R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K}$ and vice versa. The simulation succeeds because a UZS adversary can ask the oracle for an honest seed $\mathsf{seed}_{\mathsf{pk},\ell_M,\mathcal{U}}=[\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}]_1$, and does not need the updating $\mathsf{token}_{\mathsf{pk},\ell_K,\mathcal{U}}=\boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K}$ as $\boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K}$ is removed from the decryption key under ℓ_K in this game. A more formal reduction for this transition is provided in Lemma 8.

Game	iEnc	iKeyGen	Assumption
$\mathbf{G}_2$	$(\boldsymbol{x}^b_{\mathsf{pk}},\mathbf{0}^d,\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M},0)$	$(\boldsymbol{y}^b_{\mathsf{pk}},\mathbf{0}^d,\boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K},0)$	
$\mathbf{G}_{2.1}$	$(\boldsymbol{x}^b_{\mathsf{pk}},\boldsymbol{x}^0_{\mathsf{pk}},\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M},0)$	$(\boldsymbol{y}^b_{\mathsf{pk}},\mathbf{0}^d,\boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K},0)$	IND of IPE
$\mathbf{G}_{2.1.\ell_K}$ $\ell_K\in\mathcal{Q}_K$	same as in $\mathbf{G}_{2.1}$	$\ell'_K<\ell_K$: $(\mathbf{0}^d,\boldsymbol{y}^0_{\mathsf{pk}},\boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell'_K},0)$ $\ell'_K\geq\ell_K$: $(\boldsymbol{y}^b_{\mathsf{pk}},\mathbf{0}^d,\boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K},0)$	explained in Figure 7
$\mathbf{G}_3$	$(\mathbf{0}^d,\boldsymbol{x}^0_{\mathsf{pk}},\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M},0)$	$(\mathbf{0}^d,\boldsymbol{y}^0_{\mathsf{pk}},\boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K},0)$	IND of IPE

Fig. 6. Sequence of hybrids for transition from $\mathbf{G}_2$ to $\mathbf{G}_3$ in Theorem 2. All changes are made within iEnc and iDKGen algorithms for the replies of complete encryption and decryption key generation oracles respectively. The set of queried key labels in $\mathcal{Q}_K$ is assumed to be ordered.

From UZS to AWS-DDFE Without RO. To construct DDFE for attribute-weighted sums, we follow the framework that leverages the use of single-input FE for attribute-weighted sums with function-hiding inner products, denote by AWIPE, in [ATY23]. For each client $\mathsf{pk}\in\mathcal{U}$, the encryption of an AWS input $(\boldsymbol{x}_{\mathsf{pk},j},\boldsymbol{z}_{\mathsf{pk},j})_{j\in[N_{\mathsf{pk}}]}$ under $(\mathcal{U},\ell_M)$ and the generation of decryption key for a list of arithmetic branching programs $(f_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}$ can be briefly described as follows:

$$\begin{aligned}\mathsf{ct}_{\mathsf{pk}} &= \mathsf{AWIPE.Enc}_{(1^{|\mathcal{U}|}_{\mathsf{ip}},\mathsf{sk}_{\mathsf{pk}})}((\boldsymbol{x}_{\mathsf{pk},j},\boldsymbol{z}_{\mathsf{pk},j})_{j\in[N_{\mathsf{pk}}]},[\boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_M},0]_1);\\ \mathsf{dk}_{\mathsf{pk}} &= \mathsf{AWIPE.DKGen}_{(1^{|\mathcal{U}|}_{\mathsf{ip}},\mathsf{sk}_{\mathsf{pk}})}(f_{\mathsf{pk}},[\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}},0]_2)\end{aligned}$$

where $\ell_{\boldsymbol{f}}$ is a label describing $(f_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}$ and $1^{|\mathcal{U}|}_{\mathsf{ip}}$ indicates that the inner-product dimension is set up to be $|\mathcal{U}|$. The correctness of the scheme holds since

$$\begin{aligned}\mathsf{AWIPE.Dec}_{(1^{|\mathcal{U}|}_{\mathsf{ip}},\mathsf{sk}_{\mathsf{pk}})}(\mathsf{ct}_{\mathsf{pk}},\mathsf{dk}_{\mathsf{pk}}) &= \sum_{j\in[N_{\mathsf{pk}}]} f_{\mathsf{pk}}(\boldsymbol{x}_{\mathsf{pk},j})^{\top}\boldsymbol{z}_{\mathsf{pk},j}+\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}}^{\top}\cdot\boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_M}\\ &= \sum_{j\in[N_{\mathsf{pk}}]} f_{\mathsf{pk}}(\boldsymbol{x}_{\mathsf{pk},j})^{\top}\boldsymbol{z}_{\mathsf{pk},j}+\mathsf{share}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}||\ell_M}\end{aligned}$$

by the bilinear-update property of the UZS scheme.

In contrast to FH-IP-DDFE, we use UZS seeds $[\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}}]_2$ in decryption-key generation and use UZS tokens $\boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K}$ in encryption. The reason is that in the security proof, we replace the challenge

Game	Adjustment	Assumption
$\mathbf{G}^{\star}_{\ell_K} := \mathbf{G}_{2.1.\ell_K}$	see Figure 6	
$\mathbf{G}^{\star}_{\ell_K.1}$	iEnc: $(\boldsymbol{x}^b_{\mathsf{pk}}, \boldsymbol{x}^0_{\mathsf{pk}}, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}, \boldsymbol{a}^{\top}_{\mathsf{pk},\mathcal{U},\ell_M} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K} + \boldsymbol{x}^{b\,\top}_{\mathsf{pk}} \cdot \boldsymbol{y}^b_{\mathsf{pk}})$ iKeyGen: $\ell'_K < \ell_K$: $(\mathbf{0}^d, \boldsymbol{y}^0_{\mathsf{pk}}, \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell'_K}, 0)$ $\ell'_K = \ell_K$: $(\mathbf{0}^d, \mathbf{0}^d, \mathbf{0}^{\lvert\mathcal{U}\rvert}, 1)$ $\ell'_K > \ell_K$: $(\boldsymbol{y}^b_{\mathsf{pk}}, \mathbf{0}^d, \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell'_K}, 0)$	IND of IPE
$\mathbf{G}^{\star}_{\ell_K.2}$	iEnc: $(\boldsymbol{x}^b_{\mathsf{pk}}, \boldsymbol{x}^0_{\mathsf{pk}}, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}, R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K} + \boldsymbol{x}^{b\,\top}_{\mathsf{pk}} \cdot \boldsymbol{y}^b_{\mathsf{pk}})$ where $\sum_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}} R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K} = -\sum_{\mathsf{pk}\in\mathcal{C}\cap\mathcal{U}} \boldsymbol{a}^{\top}_{\mathsf{pk},\mathcal{U},\ell_M} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K}$ iKeyGen: same as in $\mathbf{G}^{\star}_{\ell_K.1}$	IND of UZS
$\mathbf{G}^{\star}_{\ell_K.3}$	iEnc: $(\boldsymbol{x}^b_{\mathsf{pk}}, \boldsymbol{x}^0_{\mathsf{pk}}, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}, R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K} + \boldsymbol{x}^{0\,\top}_{\mathsf{pk}} \cdot \boldsymbol{y}^0_{\mathsf{pk}})$ where $\sum_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}} R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K} = -\sum_{\mathsf{pk}\in\mathcal{C}\cap\mathcal{U}} \boldsymbol{a}^{\top}_{\mathsf{pk},\mathcal{U},\ell_M} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K}$ iKeyGen: same as in $\mathbf{G}^{\star}_{\ell_K.1}$	Statistics
$\mathbf{G}^{\star}_{\ell_K.4}$	iEnc: $(\boldsymbol{x}^b_{\mathsf{pk}}, \boldsymbol{x}^0_{\mathsf{pk}}, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}, \boldsymbol{a}^{\top}_{\mathsf{pk},\mathcal{U},\ell_M} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K} + \boldsymbol{x}^{0\,\top}_{\mathsf{pk}} \cdot \boldsymbol{y}^0_{\mathsf{pk}})$ iKeyGen: same as in $\mathbf{G}^{\star}_{\ell_K.1}$	IND of UZS
$\mathbf{G}^{\star}_{\ell_K+1} := \mathbf{G}_{2.1.\ell_K+1}$	see Figure 6	IND of IPE

Fig. 7. Sequence of hybrids for each transition from $\mathbf{G}_{2.1.\ell_K}$ to $\mathbf{G}_{2.1.\ell_K+1}$ in Figure 6. We denote by (ℓ_K+1) the subsequent key label of ℓ_K in the set $\mathcal{Q}_K$ of decryption key queries.

bit on messages by each pair of user set and label $(\mathcal{U}, \ell_M)$, therefore message labels ℓ_M are treated as updating labels, while key labels $\ell_{\boldsymbol{f}}$ are treated as seeding labels. Further details can be found in Section 5.

2 Preliminaries

2.1 Notations

Given any $n \in \mathbb{N}$, we denote by $[n]$ the set of integers $\{1, ..., n\}$. Given any set $\mathcal{A}$, $\mathcal{L}(\mathcal{A})$ will denote the set of finite lists of elements of $\mathcal{A}$, and $\mathcal{S}(\mathcal{A})$ will denote the set of finite subsets of $\mathcal{A}$. While both lists and sets are ordered by default, lists may contain repeated elements. We denote by $|\mathcal{A}|$ the cardinal of any finite set $\mathcal{A}$. For any vector $\boldsymbol{a} \in \mathbb{A}^n$, we denote by a_i the i-th component of $\boldsymbol{a}$. Similarly, for any matrix $\boldsymbol{A} \in \mathbb{A}^{m \times n}$, we denote by $A_{i,j}$ the component at the position (i, j) of $\boldsymbol{A}$.

We also use the following notation for condition-based function-selecting functions

$$[f_0/f_1]^{\mathsf{con}}(\mathsf{inp}) = \begin{cases} \mathsf{out} \leftarrow f_0(\mathsf{inp}) & \text{if } \mathsf{con} = 0; \\ \mathsf{out} \leftarrow f_1(\mathsf{inp}) & \text{if } \mathsf{con} = 1. \end{cases}$$

2.2 Prime Order Group.

Let GGen be a prime-order group generator, a probabilistic polynomial time (PPT) algorithm that on input the security parameter 1^λ returns a description $\mathcal{G} = (\mathbb{G}, p, P)$ of an additive cyclic group $\mathbb{G}$ of order p for a 2λ-bit prime p, whose generator is P. For $a \in \mathbb{Z}_p$, define $[a] = aP \in \mathbb{G}$ as the *implicit representation* of a in $\mathbb{G}$.

From a random element $[a] \in \mathbb{G}$, it is computationally hard to compute the value a (the discrete logarithm problem). Given $[a], [b] \in \mathbb{G}$ and a scalar $x \in \mathbb{Z}_p$, one can efficiently compute $[ax] \in \mathbb{G}$ and $[a + b] = [a] + [b] \in G$.

Definition 1 (Decisional Diffie-Hellman Assumption). *The Decisional Diffie-Hellman Assumption states that, for every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,*

$$\mathsf{Adv}_{\mathit{DDH}}(\mathcal{A}) := \left| \mathrm{P}\left[\mathcal{A}(\mathcal{G}, \mathcal{D}_b) = b \;\middle|\; \begin{array}{l} b \xleftarrow{\$} \{0,1\}, \mathcal{G} \xleftarrow{\$} \mathsf{GGen}(1^\lambda) \\ a, r, s \xleftarrow{\$} \mathbb{Z}_p, d_0 = ar, d_1 = s \\ \mathcal{D}_b = ([a], [r], [d_b]) \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

Definition 2 (m-Multi DDH Assumption [CDG+18]). *For all $\lambda \in \mathbb{N}$ and for every PPT adversary $\mathcal{A}$ running within time t, then*

$$\mathsf{Adv}_{m\text{-}\mathit{DDH}}(\mathcal{A}, t) := \left| \mathrm{P}\left[\mathcal{A}(\mathcal{G}, \mathcal{D}_b) = b \;\middle|\; \begin{array}{l} b \xleftarrow{\$} \{0,1\}, \mathcal{G} \xleftarrow{\$} \mathsf{GGen}(1^\lambda) \\ X, Y_j, Z_j \xleftarrow{\$} \mathbb{G}\ \forall j \in [m] \\ \mathcal{D}_0 = (X, (Y_j, \mathit{CDH}(X, Y_j))_{j=1}^m) \\ \mathcal{D}_1 = (X, (Y_j, Z_j)_{j=1}^m) \end{array} \right] - \frac{1}{2} \right|$$

is bounded by $\mathsf{Adv}_{\mathit{DDH}}(\mathcal{A}, t + 4m \times t_{\mathbb{G}})$, where $t_{\mathbb{G}}$ is the time for an exponentiation in $\mathbb{G}$.

2.3 Pairing Group.

Let PGGen be a pairing group generator, a PPT algorithm that on input the security parameter 1^λ returns a description $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, P_1, P_2, e)$ of asymmetric pairing groups where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are additive cyclic groups of order p for a 2λ-bit prime p, P_1 and P_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable (non-degenerate) bilinear group elements. For $s \in \{1, 2, T\}$ and $a \in \mathbb{Z}_p$, we define $[a]_s = aP_s \in \mathbb{G}_s$ as the implicit representation of a in $\mathbb{G}_s$, and then for any $0 < B < \frac{p}{2}$, we define $\mathbb{G}_s^{[-B,B]} = \{[a] \in \mathbb{G}_s : a \in [-B, B]\}$. Given $[a]_1$, $[b]_2$, one can efficiently compute $[ab]_T$ using the pairing e.

Definition 3 (Symmetric eXternal Diffie-Hellman Assumption). *The Symmetric eXternal Diffie-Hellman (SXDH) Assumption states that, in a pairing group $\mathcal{PG} \xleftarrow{\$} \mathit{PGen}(1^\lambda)$, the DDH assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$.*

2.4 Arithmetic Branching Programs

Definition 4 (Arithmetic Branching Programs (ABPs) [IW14,AGW20,ATY23]). *An arithmetic branching program $f : \mathbb{Z}_p^{n_0} \to \mathbb{Z}_p$ is defined by a prime p, a directed acyclic graph (V, E), two special vertices $v_0, v_1 \in V$, and a labeling function $\sigma : E \to \mathcal{F}^{\mathsf{Affine}}$, where $\mathcal{F}^{\mathsf{Affine}}$ consists of all affine functions $g : \mathbb{Z}_p^{n_0} \to \mathbb{Z}_p$. The size of f is the number of vertices $|V|$. Given an input $\boldsymbol{x} \in \mathbb{Z}_p^{n_0}$ to the ABP, we can assign a $\mathbb{Z}_p$ element to edge $e \in E$ by $\sigma(e)(\boldsymbol{x})$. Let P be the set of all paths from v_0 to v_1. Each element in P can be represented by a subset of E. The output of the ABP on input $\boldsymbol{x}$ is defined as $\sum_{E' \in P} \prod_{e \in E'} \sigma(e)(\boldsymbol{x})$. We can extend the definition of ABPs for functions $f : \mathbb{Z}_p^{n_0} \to \mathbb{Z}_p^{n_1}$ by evaluating each output in a coordinate-wise manner and denote such a function class by $\mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}$.*

There exists 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, so ABPs can be considered as a stronger computational model than the others.

2.5 Dynamic Decentralized Functional Encryption

Definition 5 (Dynamic Decentralized Functional Encryption). *A dynamic decentralized functional encryption scheme over a set of public keys $\mathcal{PK}$ for functionality $\mathcal{F}$: $\mathcal{L}(\mathcal{PK} \times \mathcal{K}) \times \mathcal{L}(\mathcal{PK} \times \mathcal{M}) \to \{0,1\}^*$ consists of five algorithms:*

- $\mathsf{SetUp}(1^\lambda)$*: On input a parameter 1^λ, it generates and outputs public parameters* pp*. Those parameters are implicit arguments to all the other algorithms.*
- $\mathsf{KeyGen}()$*: It generates and outputs a party's public key* $\mathsf{pk} \in \mathcal{PK}$ *and the corresponding secret key* $\mathsf{sk}_{\mathsf{pk}}$*.*
- $\mathsf{Enc}(\mathsf{sk}_{\mathsf{pk}}, m)$*: On input a party's secret key* $\mathsf{sk}_{\mathsf{pk}}$*, a value $m \in \mathcal{M}$ to encrypt, it outputs a ciphertext* $\mathsf{ct}_{\mathsf{pk},m}$*.*
- $\mathsf{DKGen}(\mathsf{sk}_{\mathsf{pk}}, k)$*: On input a party's secret key* $\mathsf{sk}_{\mathsf{pk}}$*, a key space object k, it outputs a functional decryption key* $\mathsf{dk}_{\mathsf{pk},k}$*.*
- $\mathsf{Dec}((\mathsf{dk}_{\mathsf{pk},k_{\mathsf{pk}}})_{\mathsf{pk}\in\mathcal{U}_K}, (\mathsf{ct}_{\mathsf{pk},m_{\mathsf{pk}}})_{\mathsf{pk}\in\mathcal{U}_M})$*: On input a finite list of functional decryption keys* $(\mathsf{dk}_{\mathsf{pk},k_{\mathsf{pk}}})_{\mathsf{pk}\in\mathcal{U}_K}$*, a finite list of ciphertexts* $(\mathsf{ct}_{\mathsf{pk},m_{\mathsf{pk}}})_{\mathsf{pk}\in\mathcal{U}_M}$*, where $\mathcal{U}_M, \mathcal{U}_K \in \mathcal{L}(\mathcal{PK})$ are the lists of senders and receivers respectively, it outputs a value $y \in \{0,1\}^*$.*

Correctness. *For all parameters $\lambda \in \mathbb{N}$, all polynomial size lists $\mathcal{U}_M, \mathcal{U}_K \in \mathcal{L}(\mathcal{PK})$ of public keys issued by* $\mathsf{KeyGen}()$*, $(\mathsf{pk}, k_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}_K} \in \mathcal{L}(\mathcal{PK} \times \mathcal{K})$ and $(\mathsf{pk}, m_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}_M} \in \mathcal{L}(\mathcal{PK} \times \mathcal{M})$, it holds that*

$$\Pr\left[\mathsf{Dec}((\mathsf{dk}_{\mathsf{pk},k_{\mathsf{pk}}})_{\mathsf{pk}\in\mathcal{U}_K}, (\mathsf{ct}_{\mathsf{pk},m_{\mathsf{pk}}})_{\mathsf{pk}\in\mathcal{U}_M}) = F((\mathsf{pk}, k_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}_K}, (\mathsf{pk}, m_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}_M})\right] = 1,$$

where the probability is taken over $\mathsf{pp} \leftarrow \mathsf{SetUp}(\lambda)$*,* $\mathsf{dk}_{\mathsf{pk},k_{\mathsf{pk}}} \leftarrow \mathsf{DKGen}(\mathsf{sk}_{\mathsf{pk}}, k_{\mathsf{pk}})$*, for all* $\mathsf{pk} \in \mathcal{U}_K$*, and* $\mathsf{ct}_{\mathsf{pk},m_{\mathsf{pk}}} \leftarrow \mathsf{Enc}(\mathsf{sk}_{\mathsf{pk}}, m_{\mathsf{pk}})$ *for all* $\mathsf{pk} \in \mathcal{U}_M$*.*

In this work, we assume that each user is identified by a public key pk, which it can generate on its own with the (unique) associated secret key, using KeyGen. Anyone can thus dynamically join the system, by publishing its public key.

Remark 1 (Empty lists). As in [CDSG+20], we denote by ϵ_K the empty list in $\mathcal{L}(\mathcal{PK} \times \mathcal{K})$, indicating that there is no key required. Furthermore, ϵ_M denotes the empty list in $\mathcal{L}(\mathcal{PK} \times \mathcal{M})$, indicating that there is no message required.

Definition 6 (Security for DDFE). *For* $\mathrm{xx} \in \{\mathrm{sel}, \mathrm{adt}\}$, $\mathrm{yy} \in \{\mathrm{sym}, \mathrm{asym}\}$, $\mathrm{zz} \in \{\mathrm{fh}, \mathrm{nfh}\}$*, a* xx-yy-zz-IND *security game of* DDFE *for every PPT adversary $\mathcal{A}$ is defined with access to the oracles* $\mathsf{QNewHon}$*,* QEnc*,* QDKGen *and* QCor *described below:*

- *Initialize: the challenger runs the setup algorithm* $\mathsf{pp} \leftarrow \mathsf{SetUp}(\lambda)$ *and chooses a random bit* $b \xleftarrow{\$} \{0,1\}$*. It provides* pp *to the adversary $\mathcal{A}$;*
- *Participation creation queries* $\mathsf{QNewHon}()$*: it generates* $(\mathsf{pk}, \mathsf{sk}_{\mathsf{pk}}) \leftarrow \mathsf{KeyGen}()$*, stores the association* $(\mathsf{pk}, \mathsf{sk}_{\mathsf{pk}})$ *and returns* pk *to the adversary;*
- *Challenge message queries* $\mathsf{QEnc}(\mathsf{pk}, m^0, m^1)$*: it outputs the ciphertext* $\mathsf{ct}_m \leftarrow (\mathsf{sk}, m^b)$ *where* sk *is associated with* pk*. If* pk *is not associated with any secret key, nothing is returned;*

- *Challenge key queries* $\mathsf{QDKGen}(\mathsf{pk}, k^0, k^1)$*: it outputs the decryption key* $\mathsf{dk}_k \leftarrow \mathsf{DKGen}(\mathsf{sk}, k^b)$ *where* sk *is associated with* pk*. If* pk *is not associated with any secret key, nothing is returned;*
- *Corruption queries* $\mathsf{QCor}(\mathsf{pk})$*: it outputs the secret key* sk *associated to* pk*. If* pk *is not associated with any secret key, nothing is returned;*
- *Finalize:* $\mathcal{A}$ *provides its guess* b' *on the bit* b*, and this procedure outputs the result* β *according to the analysis given below.*

Let $\mathcal{PK}$ *be the set of parties on which* $\mathsf{QNewHon}()$ *is queried,* $\mathcal{C} \subset \mathcal{PK}$ *be the set of corrupted parties,* $\mathcal{H} = \mathcal{PK} \setminus \mathcal{C}$ *be the set of honest (non-corrupted) participants at the end of the game. Finalize outputs the bit* $\beta = (b' = b)$ *if the Condition* $(*)$ *is satisfied, otherwise Finalize outputs* $\beta \xleftarrow{\$} \{0,1\}$*. Condition* $(*)$ *holds if all the following conditions hold:*

- *there do not exist two lists of messages* $(\boldsymbol{m}^0 = (\mathsf{pk}, m^0_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}_M}, \boldsymbol{m}^1 = (\mathsf{pk}, m^1_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}_M})$*, including* (ϵ_M, ϵ_M)[4]*, and two lists of keys* $(\boldsymbol{k}^0 = (\mathsf{pk}, k^0_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}_K}, \boldsymbol{k}^1 = (\mathsf{pk}, k^1_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}_K})$*, including* (ϵ_K, ϵ_K)*, such that*
  - $F(\boldsymbol{k}^0, \boldsymbol{m}^0) \neq F(\boldsymbol{k}^1, \boldsymbol{m}^1)$*;*
  - $\forall \mathsf{pk} \in \mathcal{U}_M$*,* $[\mathsf{QEnc}(\mathsf{pk}, m^0_{\mathsf{pk}}, m^1_{\mathsf{pk}})$ *was made and* $\mathsf{pk} \in \mathcal{H}]$ *or* $[m^0_{\mathsf{pk}} = m^1_{\mathsf{pk}} \in \mathcal{M}$ *and* $\mathsf{pk} \in \mathcal{C}]$*;*
  - $\forall \mathsf{pk} \in \mathcal{U}_K$*,* $[\mathsf{QDKGen}(\mathsf{pk}, k^0_{\mathsf{pk}}, k^1_{\mathsf{pk}})$ *was made and* $\mathsf{pk} \in \mathcal{H}]$ *or* $[k^0_{\mathsf{pk}} = k^1_{\mathsf{pk}} \in \mathcal{K}$ *and* $\mathsf{pk} \in \mathcal{C}]$*.*
- *when* $\mathrm{xx} = \mathrm{sel}$*: the adversary sends all its* $\mathsf{QNewHon}()$ *queries in one shot. After that it sends in one shot all* $\mathsf{QEnc}(\mathsf{pk}, m^0, m^1)$*,* $\mathsf{QDKGen}(\mathsf{pk}, k^0, k^1)$ *and* $\mathsf{QCor}(\mathsf{pk})$ *queries;*
- *when* $\mathrm{yy} = \mathrm{sym}$*: for* $\mathsf{pk} \in \mathcal{C}$*, the queries* $\mathsf{QDKGen}(\mathsf{pk}, k^0_{\mathsf{pk}}, k^1_{\mathsf{pk}})$ *and* $\mathsf{QEnc}(\mathsf{pk}, m^0_{\mathsf{pk}}, m^1_{\mathsf{pk}})$ *queries must satisfy* $k^0_{\mathsf{pk}} = k^1_{\mathsf{pk}}$ *and* $m^0_{\mathsf{pk}} = m^1_{\mathsf{pk}}$*, respectively.*
- *when* $\mathrm{zz} = \mathrm{nfh}$*: all the queries* $\mathsf{QDKGen}(\mathsf{pk}, k^0_{\mathsf{pk}}, k^1_{\mathsf{pk}})$ *must satisfy* $k^0_{\mathsf{pk}} = k^1_{\mathsf{pk}}$*.*

We say DDFE *is* xx-yy-zz-IND*-secure if given any parameter* $\lambda \in \mathbb{N}$*, for every PPT adversary* $\mathcal{A}$*, the following holds*

$$\mathsf{Adv}^{\text{xx-yy-zz}}_{\mathsf{DDFE}}(\mathcal{A}) := \left|\Pr[\beta = 1] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda).$$

Definition 7 (Function-Hiding Inner-Product DDFE over Pairing Groups). *Let* $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p)$ *be a pairing group, a function-hiding* IP-DDFE *scheme over* $\mathcal{PG}$ *is defined for a dimension* $d \in \mathbb{N}$*, a message bound* $X < \frac{p}{2}$*, a function bound* $Y < \frac{p}{2}$*, and label spaces* $(\mathcal{L}_M, \mathcal{L}_K)$ *as follows:*

$$\mathcal{K} = \left(\mathbb{G}_2^{[-Y,Y]}\right)^d \times \mathcal{S}(\mathcal{PK}) \times \mathcal{L}_K;$$
$$\mathcal{M} = \left(\mathbb{G}_1^{[-X,X]}\right)^d \times \mathcal{S}(\mathcal{PK}) \times \mathcal{L}_M.$$

Then, one has

$$F(\epsilon_K, (\mathsf{pk}, [\boldsymbol{x}_{\mathsf{pk}}]_1, \mathcal{U}_{\mathsf{pk}}, \ell_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}) = (\mathcal{U}_{\mathsf{pk}}, \ell_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}};$$
$$F((\mathsf{pk}, ([\boldsymbol{y}_{\mathsf{pk}}]_2, \mathcal{U}_{\mathsf{pk}}, \ell_{\mathsf{pk}}))_{\mathsf{pk}\in\mathcal{U}}, \epsilon_M) = (\mathcal{U}_{\mathsf{pk}}, \ell_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}};$$

for any $\mathcal{U} \in \mathcal{L}(\mathcal{PK})$ *and*

$$F((\mathsf{pk}, k_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}_K}, (\mathsf{pk}, m_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}_M}) = \begin{cases} \sum_{\mathsf{pk}\in\mathcal{U}_K} \boldsymbol{x}^\top_{\mathsf{pk}} \cdot \boldsymbol{y}_{\mathsf{pk}} & \text{if condition } (*) \\ \perp & \text{otherwise.} \end{cases}$$

FH-IP-DDFE *condition* $(*)$ *is:*

- $\mathcal{U}_K = \mathcal{U}_M = \mathcal{U}$*;*
- $\exists \ell_K \in \mathcal{L}_K$*,* $\forall \mathsf{pk} \in \mathcal{U}$*,* $k_{\mathsf{pk}} = ([\boldsymbol{y}_{\mathsf{pk}}]_2, \mathcal{U}, \ell_K)$*;*
- $\exists \ell_M \in \mathcal{L}_M$*,* $\forall \mathsf{pk} \in \mathcal{U}$*,* $m_{\mathsf{pk}} = ([\boldsymbol{x}_{\mathsf{pk}}]_1, \mathcal{U}, \ell_M)$*.*

[4] We unified the function-revealing definition of DDFE in [CDSG+20] and the function-hiding definition of DDFE in [AGT21b]. Furthermore, allowing empty lists in this condition prevents trivial wins from public information in ciphertexts and decryption keys.

We have a remark from the admissibility for FH-IP-DDFE, which will be later useful for its security proof.

Remark 2 (Admissibility for FH-IP-DDFE). The first condition of Condition $(*)$ in Definition 6 when applied to FH-IP-DDFE in Definition 7 implies that

$$\sum_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}\setminus\mathsf{pk}^\star} \boldsymbol{x}^b_{\mathsf{pk}}\cdot\boldsymbol{y}^b_{\mathsf{pk}} + \boldsymbol{x}^{\tau,b}_{\mathsf{pk}^\star}\cdot\boldsymbol{y}^b_{\mathsf{pk}^\star} = \sum_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}\setminus\mathsf{pk}^\star} \boldsymbol{x}^0_{\mathsf{pk}}\cdot\boldsymbol{y}^0_{\mathsf{pk}} + \boldsymbol{x}^{\tau,0}_{\mathsf{pk}^\star}\cdot\boldsymbol{y}^0_{\mathsf{pk}^\star}$$

for each client $\mathsf{pk}^\star\in\mathcal{H}\cap\mathcal{U}$ and each index τ of the encryption query $\mathsf{QEnc}(\mathsf{pk}^\star,(x^{\tau,0}_{\mathsf{pk}},\mathcal{U},\ell_M),(x^{\tau,1}_{\mathsf{pk}},\mathcal{U},\ell_M))$. Then one has the value

$$\Delta^b_{\mathsf{pk}^\star} := \boldsymbol{x}^{1,0}_{\mathsf{pk}^\star}\cdot\boldsymbol{y}^0_{\mathsf{pk}^\star} - \boldsymbol{x}^{1,b}_{\mathsf{pk}^\star}\cdot\boldsymbol{y}^b_{\mathsf{pk}^\star} = \boldsymbol{x}^{\tau,0}_{\mathsf{pk}^\star}\cdot\boldsymbol{y}^0_{\mathsf{pk}^\star} - \boldsymbol{x}^{\tau,b}_{\mathsf{pk}^\star}\cdot\boldsymbol{y}^b_{\mathsf{pk}^\star}$$

independent of τ for each client $\mathsf{pk}^\star$. Furthermore, $\sum_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}}\Delta^b_{\mathsf{pk}}=0$.

Remark 3 (Single-Input Inner-Product). We denote by IPE a single-input FE for function-hiding inner products, that was defined as in Definition 7 for the case $|\mathcal{PK}|=1$.

Definition 8 (One key-label restriction for FH-IP-DDFE [AGT21b]). *An* FH-IP-DDFE *scheme is* xx-yy-IND *secure as in Definition 6 under the one key-label restriction if all the adversary's queries additionally satisfy the following condition:*

- $\mathsf{QDKGen}(\mathsf{pk},([\boldsymbol{y}^0_{\mathsf{pk}}]_2,\mathcal{U}_K,\ell_K),([\boldsymbol{y}^1_{\mathsf{pk}}]_2,\mathcal{U}_K,\ell_K))$ *query can be sent only once for each pair* $(\mathsf{pk},\ell_K)\in\mathcal{H}\times\mathcal{L}_K$.

Definition 9 (Attribute-Weighted-Sum DDFE over Pairing Groups). *Let* $\mathcal{PG}=(\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,p)$ *be a pairing group, a function-revealing* AWS-DDFE *scheme over* $\mathcal{PG}$ *is defined for the class of arithmetic branching programs* $\mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}$ *and a label space* $\mathcal{L}_M$ *as follows:*

$$\mathcal{K} = \{\boldsymbol{f} := (\mathsf{pk},f_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}_K} \text{ where } f_{\mathsf{pk}}\in\mathcal{F}^{\mathsf{ABP}}_{n_0,n_1} \text{ and } \mathcal{U}_K\in\mathcal{S}(\mathcal{PK})\}$$
$$\mathcal{M} = \bigcup_{i\in\mathbb{N}}(\mathbb{Z}_p^{n_0}\times\mathbb{Z}_p^{n_1})_i\times\mathcal{S}(\mathcal{PK})\times\mathcal{L}_M.$$

Then, one has $F(\epsilon_K,(\mathsf{pk},(\boldsymbol{x}_{\mathsf{pk},j},\boldsymbol{z}_{\mathsf{pk},j})_{j\in[N_{\mathsf{pk}}]},\mathcal{U}_{\mathsf{pk}},\ell_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}) = ((\boldsymbol{x}_{\mathsf{pk},j})_{j\in[N_{\mathsf{pk}}]},\mathcal{U}_{\mathsf{pk}},\ell_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}$ *for any* $\mathcal{U}\in\mathcal{L}(\mathcal{PK})$ *and*

$$F((\mathsf{pk},k_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}_K},(\mathsf{pk},m_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}_M}) = \begin{cases} [\sum_{\mathsf{pk}\in\mathcal{U}_K}\sum_{j\in[N_{\mathsf{pk}}]} f_{\mathsf{pk}}(\boldsymbol{x}_{\mathsf{pk},j})^\top\boldsymbol{z}_{\mathsf{pk},j}]_T & \text{if condition } (*) \\ \bot & \text{otherwise.}\end{cases}$$

AWS-DDFE *condition* $(*)$ *is:*

- $\mathcal{U}_K=\mathcal{U}_M=\mathcal{U}$;
- $\exists\boldsymbol{f}\in(\mathcal{F}^{\mathsf{ABP}}_{n_0,n_1},\mathsf{pk})_{\mathsf{pk}\in\mathcal{U}}$, $\forall\mathsf{pk}'\in\mathcal{U}$, $k_{\mathsf{pk}'}=(\boldsymbol{f},\mathcal{U})$;
- $\exists\ell_M\in\mathcal{L}_M$, $\forall\mathsf{pk}\in\mathcal{U}$, $m_{\mathsf{pk}}=((\boldsymbol{x}_{\mathsf{pk},j},\boldsymbol{z}_{\mathsf{pk},j})_{j\in[N_{\mathsf{pk}}]},\mathcal{U},\ell_M)$.

Like FH-IP-DDFE, we have a remark from the admissibility for AWS-DDFE.

Remark 4 (Admissibility for AWS-DDFE). The first condition of Condition $(*)$ in Definition 5 when applied to AWS-DDFE in Definition 9 implies that

$$\sum_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}\setminus\mathsf{pk}^\star} f_{\mathsf{pk}}(\boldsymbol{x}_{\mathsf{pk},j})^\top\boldsymbol{z}^b_{\mathsf{pk},j} + f_{\mathsf{pk}^\star}(\boldsymbol{x}^\tau_{\mathsf{pk},j})^\top\boldsymbol{z}^{\tau,b}_{\mathsf{pk}^\star,j} = \sum_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}\setminus\mathsf{pk}^\star} f_{\mathsf{pk}}(\boldsymbol{x}_{\mathsf{pk},j})^\top\boldsymbol{z}^0_{\mathsf{pk},j} + f_{\mathsf{pk}^\star}(\boldsymbol{x}^\tau_{\mathsf{pk},j})^\top\boldsymbol{z}^{\tau,0}_{\mathsf{pk}^\star,j}$$

for each client $\mathsf{pk}^\star\in\mathcal{H}\cap\mathcal{U}$, each τ-indexed encryption query $\mathsf{QEnc}(\mathsf{pk}^\star,(\hat{x}^{\tau,0}_{\mathsf{pk}},\mathcal{U},\ell_M),(\hat{x}^{\tau,1}_{\mathsf{pk}},\mathcal{U},\ell_M))$ where $\hat{x}^{\tau,\gamma}_{\mathsf{pk}} := (\boldsymbol{x}^\tau_{\mathsf{pk}},\boldsymbol{z}^{\tau,\gamma}_{\mathsf{pk}})$ for $\gamma\in\{0,1\}$. Then one has the value

$$\Delta^b_{\mathsf{pk}^\star} := f_{\mathsf{pk}^\star}(\boldsymbol{x}^\tau_{\mathsf{pk}^\star,j})^\top\boldsymbol{z}^{\tau,0}_{\mathsf{pk}^\star,j} - f_{\mathsf{pk}^\star}(\boldsymbol{x}^\tau_{\mathsf{pk}^\star,j})^\top\boldsymbol{z}^{\tau,b}_{\mathsf{pk}^\star,j} = f_{\mathsf{pk}^\star}(\boldsymbol{x}^1_{\mathsf{pk}^\star,j})^\top\boldsymbol{z}^{1,0}_{\mathsf{pk}^\star,j} - f_{\mathsf{pk}^\star}(\boldsymbol{x}^1_{\mathsf{pk}^\star,j})^\top\boldsymbol{z}^{1,b}_{\mathsf{pk}^\star,j}$$

independent of τ for each client $\mathsf{pk}^\star$. Furthermore, $\sum_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}}\Delta^b_{\mathsf{pk}}=0$.

Definition 10 (Single-Input FE for AWSw/IP over Pairing Groups). *Consider the case* $|\mathcal{PK}| = 1$ *in Definition 5 for the single-input setting, and let* $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p)$ *be a pairing group, an* FE *for attribute-weighted sums with function-hiding inner product scheme over* $\mathcal{PG}$ *is defined for the class of arithmetic branching programs* $\mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}$ *as follows:*

$$\mathcal{K} = \mathcal{F}^{\mathsf{ABP}}_{n_0,n_1} \times \mathbb{G}_2^d$$
$$\mathcal{M} = \bigcup_{i\in\mathbb{N}} (\mathbb{Z}_p^{n_0} \times \mathbb{Z}_p^{n_1})_i \times \mathbb{G}_1^d.$$

Then, one has

$$F(\epsilon_K, ((\boldsymbol{x}_j, \boldsymbol{z}_j)_{j\in[N]}, [\boldsymbol{s}]_1)) = ((\boldsymbol{x}_j)_{j\in[N]});$$
$$F((f, [\boldsymbol{t}]_2), \epsilon_M) = f;$$

and

$$F((f, [\boldsymbol{t}]_2), ((\boldsymbol{x}_j, \boldsymbol{z}_j)_{j\in[N]}, [\boldsymbol{s}]_1)) = [\sum_{j\in[N]} f(\boldsymbol{x}_j)^\top \boldsymbol{z}_j + \boldsymbol{s}^\top \boldsymbol{t}]_T.$$

We note that a concrete construction of FE for AWSw/IP that obtained a function-hiding security in the standard model is provided in [ATY23].

Definition 11 (All-or-Nothing Encapsulation [CDSG+20]). AoNE *is defined on messages of length* L *and label space* $\mathcal{L}$ *as follows:*

$$\mathcal{K} = \emptyset \qquad\qquad \mathcal{M} = \{0,1\}^L \times \mathcal{S}(\mathcal{PK}) \times \mathcal{L}$$

Then, $F(\epsilon_K, (\mathsf{pk}, (x_{\mathsf{pk}}, \mathcal{U}_{\mathsf{pk}}, \ell_{\mathsf{pk}}))_{\mathsf{pk}\in\mathcal{U}}) = (\mathcal{U}_{\mathsf{pk}}, \ell_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}$ *for any* $\mathcal{U} \in \mathcal{L}(\mathcal{PK})$ *and*

$$F(\epsilon_K, (\mathsf{pk}, m_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}_M}) = \begin{cases} (\mathsf{pk}, x_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}_M} & \text{if condition } (*) \\ \perp & \text{otherwise.} \end{cases}$$

and AoNE *condition* $(*)$ *is:* $\exists \ell \in \mathcal{L}, \forall \mathsf{pk} \in \mathcal{U}_M, m_{\mathsf{pk}} = (x_{\mathsf{pk}}, \mathcal{U}_M, \ell)$.

We note that a concrete AoNE construction that is secure in the standard model is provided in [CDSG+20].

2.6 Pseudorandom Functions (PRF)

Definition 12 (PRF). *A* PRF *from input space* $\mathcal{X}$ *to output space* $\mathcal{Y}$ *is secure if for any security parameter* $\lambda \in \mathbb{N}$*, and for every PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\mathsf{Adv}_{\mathsf{PRF},\mathcal{A}}(\lambda) := \left| \mathrm{P}\left[\mathcal{A}^{\mathcal{O}^b_{\mathsf{PRF}}(\cdot)}(1^\lambda) = b \;\middle|\; \begin{array}{l} K \xleftarrow{\$} \{0,1\}^\lambda, b \xleftarrow{\$} \{0,1\} \\ \forall \ell \in \mathcal{X}: \\ \quad \mathcal{O}^0_{\mathsf{PRF}}(\ell) := \mathsf{PRF}_K(\ell) \\ \quad \mathcal{O}^1_{\mathsf{PRF}}(\ell) := \mathsf{RF}(\ell) \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda)$$

where $\mathcal{O}^b_{\mathsf{PRF}}(\cdot)$ *is an oracle depending on the challenge bit* b *and* RF *is a random function computed on the fly.*

2.7 Non-Interactive Key Exchange (NIKE)

Definition 13 (Non-Interactive Key Exchange). *A* NIKE *scheme consists of three PPT algorithms:*

- $\mathsf{SetUp}(\lambda)$: *On input a security parameter* λ*, it outputs public parameters* pp*. Those parameters are implicit arguments to all the other algorithms;*
- $\mathsf{KeyGen}()$*: It generates and outputs a party's public key* $\mathsf{pk} \in \mathcal{PK}$ *and the corresponding secret key* $\mathsf{sk}_{\mathsf{pk}}$*;*
- $\mathsf{SharedKey}(\mathsf{pk}, \mathsf{sk}_{\mathsf{pk}'})$*: On input a public key and a secret key corresponding to a different public key, it deterministically outputs a shared key* K*.*

Correctness. For all security parameters $\lambda \in \mathbb{N}$, it holds that

$$\Pr[\mathsf{SharedKey}(\mathsf{pk}, \mathsf{sk}_{\mathsf{pk}'}) = \mathsf{SharedKey}(\mathsf{pk}', \mathsf{sk}_{\mathsf{pk}})] = 1,$$

where the probability is taken over $\mathsf{pp} \leftarrow \mathsf{SetUp}(\lambda)$, $(\mathsf{pk}, \mathsf{sk}_{\mathsf{pk}}) \leftarrow \mathsf{KeyGen}()$, $(\mathsf{pk}', \mathsf{sk}'_{\mathsf{pk}}) \leftarrow \mathsf{KeyGen}()$.

Definition 14 (Security for NIKE). *No adversary A should be able to win the following security game against a challenger $\mathcal{C}$, with unlimited and adaptive access to the oracles* QNewHon, QRev, QTest, *and* QCor *described below:*

- *Initialize: the challenger runs the setup algorithm* $\mathsf{pp} \leftarrow \mathsf{SetUp}(\lambda)$ *and chooses a random bit* $b \xleftarrow{\$} \{0,1\}$. *It initializes the set $\mathcal{H}$ of honest participants to $\emptyset$. It provides* pp *to the adversary $\mathcal{A}$;*
- *Participant creation queries* $\mathsf{QNewHon}()$*: it generates* $(\mathsf{pk}, \mathsf{sk}_{\mathsf{pk}}) \leftarrow \mathsf{KeyGen}()$, *stores the association* $(\mathsf{pk}, \mathsf{sk}_{\mathsf{pk}})$ *in the set $\mathcal{H}$ of honest keys, and returns* pk *to the adversary;*
- *Reveal queries* $\mathsf{QRev}(\mathsf{pk}, \mathsf{pk}')$ *: it requires at least one of* pk *and* pk' *be in $\mathcal{H}$. Assume* $\mathsf{pk} \in \mathcal{H}$, *then it returns* $\mathsf{SharedKey}(\mathsf{pk}', \mathsf{sk}_{\mathsf{pk}})$*;*
- *Test queries* $\mathsf{QTest}(\mathsf{pk}, \mathsf{pk}')$*: it requires that both* pk *and* pk' *were generated via* QNewHon.
  - *if $b = 0$, it returns* $\mathsf{SharedKey}(\mathsf{pk}', \mathsf{sk}_{\mathsf{pk}})$*;*
  - *if $b = 1$, it returns a (uniformly) random value and stores the value so it can consistently answer further queries to* $\mathsf{QTest}(\mathsf{pk}, \mathsf{pk}')$ *or* $\mathsf{QTest}(\mathsf{pk}', \mathsf{pk})$.
- *Corruption queries* $\mathsf{QCor}(\mathsf{pk})$*: it recovers the secret key* sk *associated to* pk *from $\mathcal{H}$ and returns it, then removes* $(\mathsf{pk}, \mathsf{sk})$ *from $\mathcal{H}$. If* pk *is not associated with any secret key (i.e. it is not in $\mathcal{H}$), then nothing is returned;*
- *Finalize: $\mathcal{A}$ provides its guess b' on the bit b, and this procedure outputs the result β according to the analysis given below.*

Finalize outputs the bit $\beta = (b' = b)$ unless a QCor *query was made for any public key which was involved in a query to* QTest, *or a* QRev *query was made for a pair of public keys which was also involved in a* QTest *query, in which case a random bit β is returned. We say* NIKE *is secure if given any parameter $\lambda \in \mathbb{N}$, for every PPT adversary $\mathcal{A}$, the following holds:*

$$\mathsf{Adv}_{\mathsf{NIKE}}(\mathcal{A}) = |\Pr[\beta = 1] - 1/2| \leq \mathsf{negl}(\lambda).$$

3 Updatable Pseudorandom Zero Sharing

In this section, we provide a definition and a security model for Updatable Pseudorandom Zero-Sharing. A construction in DDH groups will be provided and supports the bilinear update property, which serves as a core building blocks for later pairing-based DDFE constructions.

3.1 Definition

Definition 15 (Updatable Pseudorandom Zero Sharing). *Given a set of users $\mathcal{PK}$, a seeding label space $\mathcal{L}_S$ and an updating label space $\mathcal{L}_U$, an updatable pseudorandom zero sharing scheme* UZS *over a group $(\mathbb{A}, +)$ consists of six algorithms:*

- $\mathsf{SetUp}(\lambda)$*: On input a security parameter λ, it outputs public parameters* pp. *Those parameters are implicit arguments to all the other algorithms.*
- $\mathsf{KeyGen}()$*: It outputs a party's public key* $\mathsf{pk} \in \mathcal{PK}$ *and the corresponding secret key* $\mathsf{sk}_{\mathsf{pk}}$.
- $\mathsf{SeedGen}(\mathsf{sk}_{\mathsf{pk}}, \mathcal{U}, \ell_s)$*: On input a secret key* $\mathsf{sk}_{\mathsf{pk}}$, *a user set* $\mathcal{U} \in \mathcal{S}(\mathcal{PK})$, *and a seeding label* $\ell_s \in \mathcal{L}_S$, *it outputs a seed* $\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_s}$ *if* $\mathsf{pk} \in \mathcal{U}$. *Otherwise, it outputs* $\perp$.
- $\mathsf{TokGen}(\mathsf{sk}_{\mathsf{pk}}, \mathcal{U}, \ell_u)$*: On input a secret key* $\mathsf{sk}_{\mathsf{pk}}$, *a user set* $\mathcal{U} \in \mathcal{S}(\mathcal{PK})$, *and an updating label* $\ell_u \in \mathcal{L}_U$, *it outputs a token* $\mathsf{token}_{\mathsf{pk},\mathcal{U},\ell_u}$ *if* $\mathsf{pk} \in \mathcal{U}$. *Otherwise, it outputs* $\perp$.
- $\mathsf{SeedUpt}(\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_s}, \mathsf{token}_{\mathsf{pk},\mathcal{U},\ell_u})$*: On input a seed* $\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_s}$ *and a token* $\mathsf{token}_{\mathsf{pk},\mathcal{U},\ell_u}$, *it deterministically outputs a seed* $\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_s||\ell_u}$.
- $\mathsf{ShareEval}(\mathsf{seed}_{\mathsf{pk},\ell})$*: On input a seed* $\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell}$ *for* $\ell \in \mathcal{L}_S \cup (\mathcal{L}_S \times \mathcal{L}_U)$, *it deterministically outputs a share* $\mathsf{share}_{\mathsf{pk},\mathcal{U},\ell}$.

Correctness. *For any security parameter $\lambda \in \mathbb{N}$, any $\ell_s \in \mathcal{L}_S$, any $\ell_u \in \mathcal{L}_U$, any $\mathcal{U} \in \mathcal{S}(\mathcal{PK})$, then it holds that*

$$\Pr\left[\sum_{\mathsf{pk}\in\mathcal{U}} \mathsf{share}_{\mathsf{pk},\mathcal{U},\ell_s} = \sum_{\mathsf{pk}\in\mathcal{U}} \mathsf{share}_{\mathsf{pk},\mathcal{U},\ell_s||\ell_u} = 0_{\mathbb{A}}\right] = 1$$

where the probability is taken over $\mathsf{pp} \leftarrow \mathsf{SetUp}(\lambda)$ *and over the following algorithms:* $\forall \mathsf{pk} \in \mathcal{U}$, $(\mathsf{sk}_{\mathsf{pk}}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}()$, $\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_s} \leftarrow \mathsf{SeedGen}(\mathsf{sk}_{\mathsf{pk}},\mathcal{U},\ell_s)$, $\mathsf{token}_{\mathsf{pk},\mathcal{U},\ell_u} \leftarrow \mathsf{TokGen}(\mathsf{sk}_{\mathsf{pk}},\mathcal{U},\ell_u)$, $\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_s||\ell_u} \leftarrow \mathsf{SeedUpt}(\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_s},\mathsf{token}_{\mathsf{pk},\mathcal{U},\ell_u})$, $\mathsf{share}_{\mathsf{pk},\mathcal{U},\ell_s} \leftarrow \mathsf{ShareEval}(\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_s})$, $\mathsf{share}_{\mathsf{pk},\mathcal{U},\ell_s||\ell_u} \leftarrow \mathsf{ShareEval}(\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_s||\ell_u})$.

Definition 16 (Correlated Pseudorandomness for UZS). *For* $\mathrm{xx} \in \{\mathrm{otu}, \mathrm{any}\}$, $\mathrm{yy} \in \{\mathrm{sta}, \mathrm{adt}\}$, *a* xx-yy-IND *security game of* UZS *for every PPT adversary $\mathcal{A}$ is defined with access to the oracles* $\mathsf{QNewHon}, \mathsf{QCor}, \mathsf{QTokGen}, \mathsf{QSeedGen},$ *and* QShare *described below:*

- *Initialize: the challenger runs the setup algorithm* $\mathsf{pp} \leftarrow \mathsf{SetUp}(\lambda)$ *and chooses a random bit* $b \xleftarrow{\$} \{0,1\}$. *It initializes the sets $\mathcal{H}$ and $\mathcal{C}$ of honest participants and corrupted participants respectively to $\emptyset$, and provides* pp *to the adversary $\mathcal{A}$;*
- *Participant creation queries* $\mathsf{QNewHon}()$*: it generates* $(\mathsf{pk}, \mathsf{sk}_{\mathsf{pk}}) \leftarrow \mathsf{KeyGen}()$ *to simulate a new participant, stores the association* $(\mathsf{pk}, \mathsf{sk}_{\mathsf{pk}})$ *in the set $\mathcal{H}$, and returns* pk *to the adversary;*
- *Corruption queries* $\mathsf{QCor}(\mathsf{pk})$*: it moves the association* $(\mathsf{pk}, \mathsf{sk}_{\mathsf{pk}})$ *from $\mathcal{H}$ to $\mathcal{C}$ and returns the secret key* sk. *If* pk *is not associated with any secret key in $\mathcal{H}$, then nothing is returned;*
- *Seed generation queries* $\mathsf{QSeedGen}(\mathsf{pk},\mathcal{U},\ell_s)$*: it returns* $\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_s} \leftarrow \mathsf{SeedGen}(\mathsf{sk},\mathcal{U},\ell_s)$ *to the adversary. If* $[\mathsf{pk}$ *is not associated with any secret key in either $\mathcal{H}$ or $\mathcal{C}$]* *or* $[\mathsf{pk} \notin \mathcal{U}]$ *or* $[\ell_s \notin \mathcal{L}_S]$, *nothing is returned;*
- *Token generation queries* $\mathsf{QTokGen}(\mathsf{pk},\mathcal{U},\ell_u)$*: it returns* $\mathsf{token}_{\mathsf{pk},\mathcal{U},\ell_u} \leftarrow \mathsf{TokGen}(\mathsf{sk},\mathcal{U},\ell_u)$ *to the adversary. If* $[\mathsf{pk}$ *is not associated with any secret key in either $\mathcal{H}$ or $\mathcal{C}$]* *or* $[\mathsf{pk} \notin \mathcal{U}]$ *or* $[\ell_u \notin \mathcal{L}_U]$, *nothing is returned;*
- *Challenge share queries* $\mathsf{QShare}(\mathcal{U},\ell)$*: if* $[\ell \notin \mathcal{L}_S \cup (\mathcal{L}_S \times \mathcal{L}_U)]$ *and* $[|\mathcal{H} \cap \mathcal{U}| < 2]$, *nothing is returned. We define the following share distributions:*
  - *A pseudorandom share generation algorithm* $\mathsf{PShareGen}(\mathcal{S},\mathcal{U},\ell)$*: it outputs $\perp$ if $\mathcal{S} \not\subset \mathcal{U}$, the challenger generates* $\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell}$ *as follows:*
    - *if* $\ell = \ell_s \in \mathcal{L}_S$*:* $\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_s} = \mathsf{QSeedGen}(\mathsf{pk},\mathcal{U},\ell_s)$;
    - *if* $\ell = \ell_s||\ell_u \in \mathcal{L}_S \times \mathcal{L}_U$*:* $\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_s||\ell_u} = \mathsf{SeedUpt}(\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_s},\mathsf{token}_{\mathsf{pk},\mathcal{U},\ell_u})$ *where* $\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_s} = \mathsf{QSeedGen}(\mathsf{pk},\mathcal{U},\ell_s)$ *and* $\mathsf{token}_{\mathsf{pk},\mathcal{U},\ell_u} = \mathsf{QTokGen}(\mathsf{pk},\mathcal{U},\ell_u)$.

    and outputs
    $$\mathsf{share}_{\mathsf{pk},\mathcal{U},\ell} \leftarrow \mathsf{ShareEval}(\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell})$$
    for all $\mathsf{pk} \in \mathcal{S}$.
  - *A uniformly correlated random distribution for any* $(\mathcal{S},\mathcal{U},\ell)$ *where* $\mathcal{S} \subset \mathcal{U}$
    $$\mathcal{R}_{\mathcal{S},\mathcal{U},\ell} := \left\{ (s_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{S}} \xleftarrow{\$} \mathbb{A}^{|\mathcal{S}|} \mid \sum_{\mathsf{pk}\in\mathcal{S}} s_{\mathsf{pk}} = -\sum_{\mathsf{pk}\in\mathcal{U}\setminus\mathcal{S}} t_{\mathsf{pk}} \right\}$$
    where $(t_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}\setminus\mathcal{S}} = \mathsf{PShareGen}(\mathcal{U}\setminus\mathcal{S},\mathcal{U},\ell)$.

  Then,
  - *if $b = 0$, the challenger returns* $(\mathsf{share}_{\mathsf{pk},\mathcal{U},\ell})_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}} \leftarrow \mathsf{PShareGen}(\mathcal{H}\cap\mathcal{U},\mathcal{U},\ell)$ *to the adversary;*
  - *if $b = 1$, the challenger returns* $(\mathsf{share}_{\mathsf{pk},\mathcal{U},\ell})_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}} \leftarrow \mathcal{R}_{\mathcal{H}\cap\mathcal{U},\mathcal{U},\ell}$ *to the adversary;*
- *Finalize: $\mathcal{A}$ provides its guess b' on the bit b, and this procedure outputs the result β of the game, according to the analysis given below which aims at preventing trivial wins.*

Let $\mathcal{H}$ and $\mathcal{C}$ be the sets of honest users and corrupted users at the end of the game respectively. Finalize outputs the bit $\beta = (b' = b)$ if the Condition $(*)$ is satisfied, otherwise Finalize outputs $\beta \xleftarrow{\$} \{0,1\}$. *Condition $(*)$ holds if all the following conditions hold:*

- *there are no* $\mathsf{QCor}(\mathsf{pk})$ *queries after a* $\mathsf{QShare}(\mathcal{U},\ell)$ *query was made;*
- *there does not exist $\ell_s \in \mathcal{L}_S$ such that a* $\mathsf{QShare}(\mathcal{U},\ell_s)$ *query and a* $\mathsf{QSeedGen}(\mathsf{pk},\ell_s)$ *query are sent;*
- *there does not exist* $(\ell_s,\ell_u) \in \mathcal{L}_S \times \mathcal{L}_U$ *such that a* $\mathsf{QShare}(\mathcal{U},\ell_s||\ell_u)$ *query and a* $\mathsf{QTokGen}(\mathsf{pk},\ell_u)$ *query are sent;*

- *when* xx = otu*: for any* $\mathcal{U}$*, the queries of the form* $\mathsf{QShare}(\mathcal{U}, \ell_s||\cdot)$ *can be sent for only one* $\ell_u \in \mathcal{L}_U$*.*
- *when* yy = sta*: the adversary sends all its* $\mathsf{QNewHon}()$ *queries in one shot. After that it sends in one shot all* $\mathsf{QCor}(\mathsf{pk})$ *queries.*

We say UZS *is* xx-yy-IND*-secure if given any parameter* $\lambda \in \mathbb{N}$*, for every PPT adversary* $\mathcal{A}$*, the following holds*

$$\mathsf{Adv}^{\text{xx-yy}}_{\mathsf{UZS}}(\mathcal{A}) = \left|\Pr[\beta = 1] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda).$$

The security model described above captures the security for a standard pseudorandom zero-sharing scheme: the adversary has full access to the oracles and can win the game by only distinguishing the pseudorandom non-updated shares from the correlated random ones. Additionally, the last condition in $(*)$ does not prevent the adversary from querying the non-updated challenge shares that result in the updated challenge ones. This intuitively means that even when the adversary has access to the non-updated shares, the updated shares remain indistinguishable from a correlated random distribution.

On the other hand, extending a standard pseudorandom zero-sharing scheme over $\mathcal{L} = \mathcal{L}_S \times \mathcal{L}_U$ to an updatable one can be achieved by using its share generation algorithm to create new shares on concatenated labels of the form $(\ell_s||\ell_u)$. However, this straightforward approach may require operations that are more complex than those allowed in pairing groups. Therefore, we specify a more pairing-friendly property for the updating algorithm of an UZS scheme.

Definition 17 (Bilinear Update). *An updatable pseudorandom zero sharing scheme* $\mathsf{UZS} = (\mathsf{SetUp}, \mathsf{KeyGen}, \mathsf{SeedGen}, \mathsf{TokGen}, \mathsf{SeedUpt}, \mathsf{ShareEval})$ *of security parameter* λ *is said to satisfy the bilinear update property if each seed is of the form* $[\boldsymbol{a}] \in \mathbb{A}^{\rho(\lambda)}$*, each token is of the form* $\boldsymbol{b} \in \mathbb{Z}^{\rho(\lambda)}$ *where* ρ *is a* λ*-dependent parameter, and for any* $\mathsf{pk} \in \mathcal{PK}$*, any* $\ell_s \in \mathcal{L}_S$*, and any* $\ell_u \in \mathcal{L}_U$*, it holds that*

$$[\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_s}] \odot_\lambda \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_u} = \mathsf{SeedUpt}([\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_s}], \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_u}),$$

where $\odot_\rho : \mathbb{A}^\rho \times \mathbb{Z}^\rho \to \mathbb{A}^\rho$ *is an entry-wise bilinear map.*

3.2 Construction in DDH Groups

Let $\mathsf{PRF} : \{0,1\}^* \to \mathbb{Z}_p$ be a pseudorandom function, $\mathsf{NIKE} = (\mathsf{SetUp}, \mathsf{KeyGen}, \mathsf{SharedKey})$ be a non-interactive key exchange protocol, $\mathcal{L}_S$ be a seeding label space, and $\mathcal{L}_U$ be an updating label space. A construction of UZS is described in Figure 8.

Correctness. Given $\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow \mathsf{SetUp}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}_{\mathsf{pk}}) \leftarrow \mathsf{KeyGen}()$ $\forall \mathsf{pk} \in \mathcal{PK}$, $\mathcal{U} \in \mathcal{S}(\mathcal{PK})$, any labels $\ell_s \in \mathcal{L}_S$ and $\ell_u \in \mathcal{L}_U$, from the above scheme, one has

$$\begin{aligned}
\sum_{\mathsf{pk}\in\mathcal{U}} \mathsf{share}_{\mathsf{pk},\mathcal{U},\ell_s} &= \sum_{\mathsf{pk}\in\mathcal{U}} \sum_{\mathsf{pk}'\in\mathcal{U}\setminus\{\mathsf{pk}\}} [a_{\mathsf{pk},\mathcal{U},\ell_s,\mathsf{pk}'}] \\
&= \sum_{\mathsf{pk}\in\mathcal{U}} \sum_{\mathsf{pk}'\in\mathcal{U}\setminus\{\mathsf{pk}\}} [(-1)^{\mathsf{pk}<\mathsf{pk}'} c_{\mathsf{pk},\mathsf{pk}'}] \\
&= \sum_{\substack{(\mathsf{pk},\mathsf{pk}')\in\mathcal{U}^2 \\ \mathsf{pk}\neq\mathsf{pk}'}} [(-1)^{\mathsf{pk}<\mathsf{pk}'} c_{\mathsf{pk},\mathsf{pk}'} + (-1)^{\mathsf{pk}'<\mathsf{pk}} c_{\mathsf{pk}',\mathsf{pk}}] \\
&= \sum_{\substack{(\mathsf{pk},\mathsf{pk}')\in\mathcal{U}^2 \\ \mathsf{pk}\neq\mathsf{pk}'}} [0] \\
&= [0].
\end{aligned}$$

where $c_{\mathsf{pk},\mathsf{pk}'} := \mathsf{PRF}_{k_{\mathsf{pk},\mathsf{pk}'}}(\text{"}s\text{"}||\mathcal{U}||\ell_s)$ and $c_{\mathsf{pk}',\mathsf{pk}} = c_{\mathsf{pk},\mathsf{pk}'}$. Similarly, one has

$$\begin{aligned}
\sum_{\mathsf{pk}\in\mathcal{U}} \mathsf{share}_{\mathsf{pk},\mathcal{U},\ell_s||\ell_u} &= \sum_{\mathsf{pk}\in\mathcal{U}} \sum_{\mathsf{pk}'\in\mathcal{U}\setminus\{\mathsf{pk}\}} [a_{\mathsf{pk},\mathcal{U},\ell_s,\mathsf{pk}'} b_{\mathsf{pk},\mathcal{U},\ell_u,\mathsf{pk}'}] \\
&= \sum_{\mathsf{pk}\in\mathcal{U}} \sum_{\mathsf{pk}'\in\mathcal{U}\setminus\{\mathsf{pk}\}} [(-1)^{\mathsf{pk}<\mathsf{pk}'} c'_{\mathsf{pk},\mathsf{pk}'}] \\
&= \sum_{\substack{(\mathsf{pk},\mathsf{pk}')\in\mathcal{U}^2 \\ \mathsf{pk}\neq\mathsf{pk}'}} [(-1)^{\mathsf{pk}<\mathsf{pk}'} c'_{\mathsf{pk},\mathsf{pk}'} + (-1)^{\mathsf{pk}'<\mathsf{pk}} c'_{\mathsf{pk}',\mathsf{pk}}] \\
&= \sum_{\substack{(\mathsf{pk},\mathsf{pk}')\in\mathcal{U}^2 \\ \mathsf{pk}\neq\mathsf{pk}'}} [0] \\
&= [0].
\end{aligned}$$

where $c'_{\mathsf{pk},\mathsf{pk}'} := \mathsf{PRF}_{k_{\mathsf{pk},\mathsf{pk}'}}(\text{"}s\text{"}||\mathcal{U}||\ell_s)\mathsf{PRF}_{k_{\mathsf{pk},\mathsf{pk}'}}(\text{"}u\text{"}||\mathcal{U}||\ell_u)$ and $c'_{\mathsf{pk}',\mathsf{pk}} = c'_{\mathsf{pk},\mathsf{pk}'}$. □

Construction:

- $\mathsf{SetUp}(1^\lambda)$: It generates $\mathcal{G} \leftarrow \mathsf{GGen}(1^\lambda)$ and $\mathsf{NIKE.pp} \leftarrow \mathsf{NIKE.SetUp}(1^\lambda)$ and returns
$$\mathsf{pp} = (\mathcal{G}, \mathsf{NIKE.pp}, \mathsf{PRF}, \mathcal{L}_S, \mathcal{L}_U).$$
The parameters pp are implicit to other algorithms.
- $\mathsf{KeyGen}()$: Each user samples
  - NIKE keys $(\mathsf{pk}, \mathsf{NIKE.sk}_{\mathsf{pk}}) \leftarrow \mathsf{NIKE.KeyGen}()$;
  - a shared key $k_{\mathsf{pk},\mathsf{pk}'} \leftarrow \mathsf{NIKE.SharedKey}(\mathsf{pk}', \mathsf{NIKE.sk}_{\mathsf{pk}})$ for each published $\mathsf{pk}' \in \mathcal{PK}\setminus\{\mathsf{pk}\}$.

  It returns
$$(\mathsf{pk}, \mathsf{sk}_{\mathsf{pk}}) = \left(\mathsf{pk}, (\mathsf{NIKE.sk}_{\mathsf{pk}}, (k_{\mathsf{pk},\mathsf{pk}'})_{\mathsf{pk}'\in\mathcal{PK}\setminus\mathsf{pk}})\right).$$
- $\mathsf{SeedGen}(\mathsf{sk}_{\mathsf{pk}}, (\mathcal{U}, \ell_s))$: It computes
$$\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_s} = \left((-1)^{\mathsf{pk}<\mathsf{pk}'} \mathsf{PRF}_{k_{\mathsf{pk},\mathsf{pk}'}}(\text{"}s\text{"}||\mathcal{U}||\ell_s)\right)_{\mathsf{pk}'\in\mathcal{U}\setminus\{\mathsf{pk}\}},$$
and returns $\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_s} = [\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_s}]$. If $\mathsf{pk} \notin \mathcal{U}$, it returns $\perp$.
- $\mathsf{TokGen}(\mathsf{sk}_{\mathsf{pk}}, (\mathcal{U}, \ell_u))$: It computes
$$\boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_u} = \left(\mathsf{PRF}_{k_{\mathsf{pk},\mathsf{pk}'}}(\text{"}u\text{"}||\mathcal{U}||\ell_u)\right)_{\mathsf{pk}'\in\mathcal{U}\setminus\{\mathsf{pk}\}},$$
and returns $\mathsf{token}_{\mathsf{pk},\mathcal{U},\ell_u} = \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_u}$. If $\mathsf{pk} \notin \mathcal{U}$, it returns $\perp$.
- $\mathsf{SeedUpt}(\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_s}, \mathsf{token}_{\mathsf{pk},\mathcal{U},\ell_u})$: It parses
  - $\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_s} = [\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_s}]$,
  - $\mathsf{token}_{\mathsf{pk},\mathcal{U},\ell_u} = \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_u}$,
  - $\rho = |\mathcal{U}| - 1$;

  and returns
$$\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_s||\ell_u} = [\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_s}] \odot_\rho \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_u}.$$
- $\mathsf{ShareEval}(\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell})$: It parses $\rho = |\mathcal{U}| - 1$, $\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell} = [\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_s}] \in \mathbb{G}^\rho$, and returns
$$\mathsf{share}_{\mathsf{pk},\mathcal{U},\ell} = \sum_{\mathsf{pk}'\in\mathcal{U}\setminus\{\mathsf{pk}\}} [a_{\mathsf{pk},\mathcal{U},\ell_s,\mathsf{pk}'}].$$

Fig. 8. Updatable Pseudorandom Zero-Sharing in DDH groups

Remark 5 (Bilinear Update). The UZS scheme in the above construction satisfies the bilinear update property in Definition 17.

3.3 Security Analysis

Theorem 1 (Indistinguishability for UZS). *If* NIKE *is a* IND*-secure non-interactive key exchange protocol, and the DDH assumption holds in* $\mathbb{G}$*, then the* UZS *scheme constructed in Section 3.2 is* otu-sta-IND *secure (as defined in Definition 16) in the standard model.*

Proof. In the static corruption game, we can fix a $\mathcal{PK}$ to be the set of parties generated by $\mathsf{QNewHon}()$, $\mathcal{C}$ to be the set of corrupted parties and $\mathcal{H} = \mathcal{PK} \setminus \mathcal{C}$ to be the set of honest parties. Let q_{QNewHon} and q_{QShare} be the number of QNewHon and QShare queries respectively. We proceed via a hybrid argument by using the games described in Figure 9. In this argument, the game $\mathbb{G}_0$ corresponds to the otu-sta-IND security game as defined in Definition 16, and the game $\mathbb{G}_3$ corresponds to the case where the adversary's advantage is 0 since there is no challenge bit b. Given $\lambda \in \mathbb{N}$, we denote by Adv_i the advantage of a PPT adversary $\mathcal{A}$ running in time t in each game $\mathbf{G}_i$, and $\mathsf{Adv}_{\mathrm{xx}}$ be the best advantage of any PPT adversary running in time t against the primitive xx that is setup with λ.

Game $\mathbf{G}_1$: The change is that for each $(\mathsf{pk}, \mathsf{pk}') \in \mathcal{H}^2$, the challenger uses (uniformly) random shared keys $k_{\mathsf{pk},\mathsf{pk}'} \xleftarrow{\$} \mathbb{Z}_p$ instead of $k_{\mathsf{pk},\mathsf{pk}'} \leftarrow \mathsf{NIKE.SharedKey}(\mathsf{pk}', \mathsf{sk}_{\mathsf{pk}})$ in generating answers to $\mathsf{QSeedGen}(\mathsf{pk}^\star, \mathcal{U}, \ell_u)$, $\mathsf{QTokGen}(\mathsf{pk}^\star, \mathcal{U}, \ell_s)$ for $\mathsf{pk}^\star \in \{\mathsf{pk}, \mathsf{pk}'\}$, and $\mathsf{QShare}(\mathcal{U}, \ell)$ queries. The indistinguishability is implied by the security of the non-interactive key exchange protocol, given in Lemma 1.

Game $\mathbf{G}_2$: The change is that for each $(\mathsf{pk}, \mathsf{pk}') \in \mathcal{H}^2$, the challenger uses a random function $\mathsf{RF}_{\mathsf{pk},\mathsf{pk}'} = \mathsf{RF}_{\mathsf{pk}',\mathsf{pk}}$ instead of $\mathsf{PRF}_{k_{\mathsf{pk},\mathsf{pk}'}}$ in generating answers to $\mathsf{QSeedGen}(\mathsf{pk}^\star, \mathcal{U}, \ell_u)$, $\mathsf{QTokGen}(\mathsf{pk}^\star, \mathcal{U}, \ell_s)$ for $\mathsf{pk}^\star \in \{\mathsf{pk}, \mathsf{pk}'\}$, and $\mathsf{QShare}(\mathcal{U}, \ell)$ queries. The indistinguishability is implied by the security of the pseudorandom functions, given in Lemma 2.

Game $\mathbf{G}_3$: The change is that for each $\mathsf{QShare}(\mathcal{U}, \ell)$ query, the challenger answers independently from the bit b by sampling $(\mathsf{share}_{\mathsf{pk},\ell})_{\mathsf{pk} \in \mathcal{H} \cap \mathcal{U}} \xleftarrow{\$} \mathcal{R}_{\mathcal{H}\cap\mathcal{U},\mathcal{U},\ell}$, where the distribution $\mathcal{R}_{\mathcal{H}\cap\mathcal{U},\mathcal{U},\ell}$ is defined in the QShare oracle in Definition 16. The indistinguishability is implied by the multi-DDH assumption, given in Lemma 3.

From the transitions above, one completes the theorem by having

$$\begin{aligned}\mathsf{Adv}^{\text{otu-sta}}_{\mathsf{UZS}} \leq & \mathsf{Adv}_{\mathsf{NIKE}} + q_{\mathsf{QNewHon}}(q_{\mathsf{QNewHon}} - 1) \cdot \mathsf{Adv}_{\mathsf{PRF}} \\ & + \frac{1}{2} q_{\mathsf{QNewHon}}(q_{\mathsf{QNewHon}} - 1) q_{\mathsf{QShare}} \cdot \mathsf{Adv}_{\mathsf{DDH}}(t + 4q_{\mathsf{QShare}} \times t_{\mathbb{G}}).\end{aligned}$$

where $t_{\mathbb{G}}$ is the time for an exponentiation in $\mathbb{G}$. □

Lemma 1 (UZS: Transition from $\mathbf{G}_0$ to $\mathbf{G}_1$). *For any PPT adversary* $\mathcal{A}$*, the advantage in distinguishing two games is*

$$|\mathsf{Adv}_0 - \mathsf{Adv}_1| \leq \mathsf{Adv}_{\mathsf{NIKE}}.$$

Proof. We build an adversary $\mathcal{B}$ against the IND security of NIKE from an adversary $\mathcal{A}$ that distinguishes between $\mathbf{G}_0$ and $\mathbf{G}_1$. To simulate a UZS challenger, $\mathcal{B}$ uses the NIKE oracles as follows:

- For every $\mathsf{QNewHon}()$ query, the adversary $\mathcal{B}$ returns pk from the $\mathsf{NIKE.QNewHon}()$ query to $\mathcal{A}$.
- For every $\mathsf{QCor}(\mathsf{pk})$ query, $\mathcal{B}$ obtains $\mathsf{NIKE.sk}_{\mathsf{pk}}$ from the $\mathsf{NIKE.QCor}(\mathsf{pk})$ query, and computes $k_{\mathsf{pk},\mathsf{pk}'} \leftarrow \mathsf{NIKE.SharedKey}(\mathsf{pk}', \mathsf{NIKE.sk}_{\mathsf{pk}})$ for all $\mathsf{pk}' \in \mathcal{PK}$ to complete the reply to $\mathcal{A}$.
- Instead of generating by itself the keys $(k_{\mathsf{pk},\mathsf{pk}'})_{(\mathsf{pk},\mathsf{pk}') \in \mathcal{H}^2, \mathsf{pk} \neq \mathsf{pk}'}$, the adversary $\mathcal{B}$ uses the challenge shared keys $k_{\mathsf{pk},\mathsf{pk}'}$ from the $\mathsf{NIKE.QTest}(\mathsf{pk}, \mathsf{pk}')$ queries.
- $\mathcal{B}$ outputs $\mathcal{A}$'s guess for the challenge bit $\mathsf{NIKE}.b$.

The admissibility condition $(*)$ of NIKE holds since the set of corrupted users in UZS is sent in one shot. Therefore, when $\mathsf{NIKE}.b = 0$, $\mathcal{B}$ is playing $\mathbf{G}_0$; when $\mathsf{NIKE}.b = 1$, $\mathcal{B}$ is playing $\mathbf{G}_1$. □

Lemma 2 (UZS: Transition from $\mathbf{G}_1$ to $\mathbf{G}_2$). *For any PPT adversary* $\mathcal{A}$*, the advantage in distinguishing two games is*

$$|\mathsf{Adv}_1 - \mathsf{Adv}_2| \leq q_{\mathsf{QNewHon}}(q_{\mathsf{QNewHon}} - 1) \cdot \mathsf{Adv}_{\mathsf{PRF}}.$$

$\mathbf{G}_0$, $\boxed{\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3}$:

$\mathsf{pp} \leftarrow \mathsf{SetUp}(1^\lambda)$, $b \xleftarrow{\$} \{0,1\}$
$(\mathcal{PK}, \mathsf{st}_1) \leftarrow \mathcal{A}(\mathsf{pp})$
$(\mathsf{pk})_{\mathsf{pk}\in\mathcal{PK}} \leftarrow \mathsf{QNewHon}([\mathcal{PK}])$
$(\mathcal{C}, \mathsf{st}_2) \leftarrow \mathcal{A}((\mathsf{pk})_{\mathsf{pk}\in\mathcal{PK}}, \mathsf{st}_1)$
$(\mathsf{pk}, \mathsf{sk}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{C}} \leftarrow \mathsf{QCor}([\mathcal{C}])$
$\boxed{\text{For } (\mathsf{pk}_1, \mathsf{pk}_2) \in \mathcal{H}^2 \text{ and } \mathsf{pk}_1 \neq \mathsf{pk}_2 : k_{\mathsf{pk}_1,\mathsf{pk}_2} \xleftarrow{\$} \mathbb{Z}_p}$
$b' \leftarrow \mathcal{A}^{\mathsf{QSeedGen}(\cdot,\cdot,\cdot),\mathsf{QTokGen}(\cdot,\cdot,\cdot),\mathsf{QShare}(\cdot,\cdot)}((\mathsf{pk}, \mathsf{sk}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{C}}, \mathsf{st}_2)$
Output: b' if Condition $(*)$ is satisfied, or $b' \xleftarrow{\$} \{0,1\}$ otherwise.

$\underline{\mathsf{QNewHon}()}$:
$(\mathsf{pk}, \mathsf{NIKE.sk}_{\mathsf{pk}}) \leftarrow \mathsf{NIKE.KeyGen}()$
$\forall \mathsf{pk}' \in \mathcal{PK} : k_{\mathsf{pk},\mathsf{pk}'} \leftarrow \mathsf{NIKE.SharedKey}(\mathsf{pk}', \mathsf{sk}_{\mathsf{pk}})$
Store $(\mathsf{pk}, \mathsf{sk}_{\mathsf{pk}}) = (\mathsf{pk}, \mathsf{NIKE.sk}_{\mathsf{pk}}, (k_{\mathsf{pk},\mathsf{pk}'})_{\mathsf{pk}'\in\mathcal{U}\setminus\{\mathsf{pk}\}})$
Return pk

$\underline{\mathsf{QCor}(\mathsf{pk})}$:
If $\mathsf{pk} \notin \mathcal{PK}$, return $\perp$.
Return $\mathsf{sk}_{\mathsf{pk}} = (\mathsf{NIKE.sk}_{\mathsf{pk}}, (k_{\mathsf{pk},\mathsf{pk}'})_{\mathsf{pk}'\in\mathcal{U}\setminus\{\mathsf{pk}\}})$.

$\underline{\mathsf{QSeedGen}(\mathsf{pk}, \mathcal{U}, \ell_s)}$:
$\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_s} = \left((-1)^{\mathsf{pk}<\mathsf{pk}'} \mathsf{PRF}_{k_{\mathsf{pk},\mathsf{pk}'}}(\text{"}s\text{"}||\mathcal{U}||\ell_s)\right)_{\mathsf{pk}'\in\mathcal{U}\setminus\{\mathsf{pk}\}}$
$\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_s} = \left((-1)^{\mathsf{pk}<\mathsf{pk}'} [\mathsf{PRF}_{k_{\mathsf{pk},\mathsf{pk}'}}/\mathsf{RF}_{\mathsf{pk},\mathsf{pk}'}]^{(\mathsf{pk},\mathsf{pk}')\in\mathcal{H}^2}(\text{"}s\text{"}||\mathcal{U}||\ell_s)\right)_{\mathsf{pk}'\in\mathcal{U}\setminus\{\mathsf{pk}\}}$
Return $\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_s} = [\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_s}]$.

$\underline{\mathsf{QTokGen}(\mathsf{pk}, \mathcal{U}, \ell_u)}$:
$\boldsymbol{b}_{\mathsf{pk},\ell_b} = \left(\mathsf{PRF}_{k_{\mathsf{pk},\mathsf{pk}'}}(\text{"}u\text{"}||\mathcal{U}||\ell_u)\right)_{\mathsf{pk}'\in\mathcal{U}\setminus\{\mathsf{pk}\}}$
$\boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_u} = \left([\mathsf{PRF}_{k_{\mathsf{pk},\mathsf{pk}'}}/\mathsf{RF}_{\mathsf{pk},\mathsf{pk}'}]^{(\mathsf{pk},\mathsf{pk}')\in\mathcal{H}^2}(\text{"}u\text{"}||\mathcal{U}||\ell_u)\right)_{\mathsf{pk}'\in\mathcal{U}\setminus\{\mathsf{pk}\}}$
Return $\mathsf{token}_{\mathsf{pk},\mathcal{U},\ell_u} = \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_u}$.

$\underline{\mathsf{QShare}(\mathcal{U}, \ell)}$:
Return $(\mathsf{share}_{\mathsf{pk},\mathcal{U},\ell})_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}} \leftarrow [\mathsf{PShareGen}/R]^b(\mathcal{H}\cap\mathcal{U}, \mathcal{U}, \ell)$
$(\mathsf{share}_{\mathsf{pk},\mathcal{U},\ell})_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}} \leftarrow R_{\mathcal{H}\cap\mathcal{U},\mathcal{U},\ell}$

Fig. 9. Games for the otu-sta-IND proof of UZS in Theorem 1

Proof. We proceed by using multiple hybrid games over each pair of different honest parties $(\mathsf{pk}, \mathsf{pk}') \in \mathcal{H}^2$. For each transition, we build an adversary $\mathcal{B}$ against the IND security of PRF from an adversary $\mathcal{A}$ that distinguishes two games in the transition. To simulate a UZS challenger, $\mathcal{B}$ uses the PRF oracle, instead of generating by itself the PRF key $k_{\mathsf{pk},\mathsf{pk}'}$ to handle all $\mathsf{PRF}_{k_{\mathsf{pk},\mathsf{pk}'}}$ related operations, and finally outputs $\mathcal{A}$'s guess for the challenge bit $\mathsf{PRF}.b$. Since the number of honest parties is bounded by $q_{\mathsf{QNewHon}}(q_{\mathsf{QNewHon}} - 1)$, one completes the proof. □

Lemma 3 (UZS: Transition from $\mathbf{G}_2$ to $\mathbf{G}_3$). *For any PPT adversary $\mathcal{A}$, the advantage in distinguishing two games is*

$$|\mathsf{Adv}_2 - \mathsf{Adv}_3| \leq \frac{1}{2} q_{\mathsf{QNewHon}}(q_{\mathsf{QNewHon}} - 1) q_{\mathsf{QShare}} \cdot \mathsf{Adv}_{DDH}(t + 4q_{\mathsf{QShare}} \times t_{\mathbb{G}}).$$

Proof. For every $\mathcal{U}$ queried in the form of $\mathsf{QShare}(\mathcal{U}, \cdot)$, we use multiple hybrid games that go all over the pairs in $\mathcal{HP}_{\mathcal{U}} = \{(\mathsf{pk}, \mathsf{pk}') \in (\mathcal{H} \cap \mathcal{U})^2, \mathsf{pk} \neq \mathsf{pk}'\}$. Without loss of generality, we assume that all pairs in $\mathcal{HP}_{\mathcal{U}}$ are bijectively mapped by κ to $[q_{\mathcal{U}}]$ for some integer $q_{\mathcal{U}} \geq 1$. For $i \in [q_{\mathcal{U}}]$, we define the following sequence of games as follows:

Game $\mathbf{G}_{2.\mathcal{U}.i}$: In this game, for any $\mathsf{QShare}(\mathcal{U}, \ell_s || \ell_u^\star)$ query[5], when $b = 0$, for all $\mathsf{pk} \in \mathcal{H} \cap \mathcal{U}$, the challenger computes

$$\begin{aligned} [\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_s}] &= \mathsf{QSeedGen}(\mathsf{pk}, \mathcal{U}, \ell_s), \\ \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_u^\star} &= \mathsf{QTokGen}(\mathsf{pk}, \mathcal{U}, \ell_u^\star), \\ \mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_s} &= [\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_s||\ell_u^\star}] \end{aligned}$$

where

$$\begin{cases} a_{\mathsf{pk},\mathcal{U},\ell_s||\ell_u^\star,\mathsf{pk}'} \xleftarrow{\$} \mathbb{Z}_p & \text{if } (\mathsf{pk},\mathsf{pk}') \in \mathcal{HP}_{\mathcal{U}} \text{ and } \kappa(\mathsf{pk},\mathsf{pk}') < i \\ a_{\mathsf{pk},\mathcal{U},\ell_s||\ell_u^\star,\mathsf{pk}'} \xleftarrow{\$} \mathbb{Z}_p & \text{if } (\mathsf{pk},\mathsf{pk}') \in \mathcal{HP}_{\mathcal{U}} \text{ and } \kappa(\mathsf{pk},\mathsf{pk}') = i \\ a_{\mathsf{pk},\mathcal{U},\ell_s||\ell_u^\star,\mathsf{pk}'} = [a_{\mathsf{pk},\mathcal{U},\ell_s,\mathsf{pk}'}] \cdot b_{\mathsf{pk},\mathcal{U},\ell_u,\mathsf{pk}'} & \text{if } (\mathsf{pk},\mathsf{pk}') \in \mathcal{HP}_{\mathcal{U}} \text{ and } \kappa(\mathsf{pk},\mathsf{pk}') > i \\ a_{\mathsf{pk},\mathcal{U},\ell_s||\ell_u^\star,\mathsf{pk}'} = [a_{\mathsf{pk},\mathcal{U},\ell_s,\mathsf{pk}'}] \cdot b_{\mathsf{pk},\mathcal{U},\ell_u,\mathsf{pk}'} & \text{otherwise} \end{cases}$$

and outputs $\mathsf{share}_{\mathsf{pk},\mathcal{U},\ell_s||\ell_u^\star} = \mathsf{ShareEval}(\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_s||\ell_u^\star})$. The change between $\mathbf{G}_{2.i-1}$ and $\mathbf{G}_{2.i}$ is in the $\mathsf{PShareGen}$ algorithm (defined by the QShare oracle in Definition 16) and is highlighted in gray.

For every $\mathcal{U}$, we assume that $\mathbf{G}_{2.\mathcal{U}.0}$ is the game where $\mathsf{QShare}(\mathcal{U}, \cdot)$ is the same as in $\mathbf{G}_2$. To transition from $\mathbf{G}_{2.\mathcal{U}.i-1}$ to $\mathbf{G}_{2.\mathcal{U}.i}$ for $i \in [q_{\mathcal{U}}]$, we build an adversary $\mathcal{B}$ against the multi-DDH assumption, which can be described as follows:

- To answer $\mathsf{QSeedGen}(\mathsf{pk}, \mathcal{U}, \ell_s)$ and $\mathsf{QSeedGen}(\mathsf{pk}', \mathcal{U}, \ell_s)$ queries, the adversary $\mathcal{B}$ implicitly uses $[c_{\mathsf{pk},\mathcal{U},\ell_s,\mathsf{pk}'}] := Y_{\ell_s} \xleftarrow{\$} \mathbb{G}$ from the multi-DDH when $\kappa(\mathsf{pk},\mathsf{pk}') = i$ to compute $[a_{\mathsf{pk},\mathcal{U},\ell_s,\mathsf{pk}'}] = [(-1)^{\mathsf{pk}<\mathsf{pk}'} c_{\mathsf{pk},\mathcal{U},\ell_s,\mathsf{pk}'}]$ and $[a_{\mathsf{pk}',\ell_s,\mathsf{pk}}] = [(-1)^{\mathsf{pk}'<\mathsf{pk}} c_{\mathsf{pk},\mathcal{U},\ell_s,\mathsf{pk}'}]$ respectively.
- To answer $\mathsf{QShare}(\mathcal{U}, \ell_s || \ell_u^\star)$ queries, if $b = 0$, the adversary $\mathcal{B}$ implicitly uses $[c_{\mathsf{pk},\mathcal{U},\ell_s||\ell_u^\star,\mathsf{pk}'}] := Z_{\ell_s}$ from the multi-DDH when $\kappa(\mathsf{pk},\mathsf{pk}') = i$ to compute $[a_{\mathsf{pk},\mathcal{U},\ell_s||\ell_u^\star,\mathsf{pk}'}] = [(-1)^{\mathsf{pk}<\mathsf{pk}'} c_{\mathsf{pk},\mathcal{U},\ell_s||\ell_u^\star,\mathsf{pk}'}]$ and $[a_{\mathsf{pk}',\ell_s||\ell_u^\star,\mathsf{pk}}] = [(-1)^{\mathsf{pk}'<\mathsf{pk}} c_{\mathsf{pk},\mathcal{U},\ell_s||\ell_u^\star,\mathsf{pk}'}]$ respectively.
- $\mathcal{B}$ outputs $\mathcal{A}$'s guess for the challenge bit $\mathsf{DDH}^{\mathsf{multi}}.b$.

The adversary $\mathcal{B}$ has a complete multi-DDH challenge $\mathcal{D} = (X, (Y_{\ell_s})_{\ell_s}, (Z_{\ell_s})_{\ell_s})$ where X can be (implicitly) considered as $[b_{\mathsf{pk},\ell_u^\star,\mathsf{pk}'}] \xleftarrow{\$} \mathbb{G}$.

- When $\mathsf{DDH}^{\mathsf{multi}}.b = 0$, one has $[c_{\mathsf{pk},\mathcal{U},\ell_s||\ell_u^\star,\mathsf{pk}'}] = Z_{\ell_s} = \mathsf{CDH}(X, Y_{\ell_s}) = [c_{\mathsf{pk},\mathcal{U},\ell_s,\mathsf{pk}'}] \cdot b_{\mathsf{pk},\mathcal{U},\ell_u,\mathsf{pk}'}$, which corresponds to $\mathbf{G}_{2.\mathcal{U}.i-1}$.
- When $\mathsf{DDH}^{\mathsf{multi}}.b = 1$, one has $[c_{\mathsf{pk},\mathcal{U},\ell_s||\ell_u^\star,\mathsf{pk}'}] = Z_{\mathcal{U}||\ell_s||\ell_u^\star} \xleftarrow{\$} \mathbb{Z}_p$, which corresponds to $\mathbf{G}_{2.\mathcal{U}.i}$.

[5] In the one-time-update setting, for each $\mathcal{U}$, ℓ_s can change while $\ell_u^\star$ is fixed.

Therefore, the computational gap between each $\mathbf{G}_{2.\mathcal{U}.i-1}$ and $\mathbf{G}_{2.\mathcal{U}.i}$ happens only when $b = 0$ and is bounded by $\frac{1}{2} \cdot \mathsf{Adv}_{\mathsf{DDH}}(\lambda, t + 4q_{\mathsf{QShare}} \cdot t_{\mathbb{G}})$.

The last step is to show that for the last $\mathcal{U}^\star$ queried to $\mathsf{QShare}(\cdot,\cdot)$, one has $\mathbf{G}_{2.\mathcal{U}^\star.q_{\mathcal{U}^\star}} = \mathbf{G}_3$. It suffices to describe the case $b = 0$. We note that for all $(\mathcal{U}, \ell)$ queried to $\mathsf{QShare}(\cdot,\cdot)$, one has $\mathsf{QShare}(\mathcal{U}, \ell) = (\mathsf{share}_{\mathsf{pk},\mathcal{U},\ell})_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}}$ where $\mathsf{share}_{\mathsf{pk},\mathcal{U},\ell} = \sum_{\mathsf{pk}'\in\mathcal{U}\setminus\{\mathsf{pk}\}}[a_{\mathsf{pk},\mathcal{U},\ell,\mathsf{pk}'}]$. By all transitions until $\mathbf{G}_{2.\mathcal{U}^\star.q_h}$, we have $a_{\mathsf{pk},\mathcal{U},\ell,\mathsf{pk}'} \xleftarrow{\$} \mathbb{Z}_p$ for any pair of honest users $(\mathsf{pk}, \mathsf{pk}')$, any set $\mathcal{U}$, any label ℓ. As $a_{\mathsf{pk}',\ell,\mathsf{pk}} = -a_{\mathsf{pk},\mathcal{U},\ell,\mathsf{pk}'}$, then

$$\left(\mathsf{share}^0_{\mathsf{pk},\mathcal{U}\cap\mathcal{H},\ell} := \sum_{\mathsf{pk}'\in\mathcal{H}\cap\mathcal{U}\setminus\{\mathsf{pk}\}} a_{\mathsf{pk},\mathcal{U},\ell,\mathsf{pk}'}\right)_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}}$$

are uniformly random shares of zero among users in $\mathcal{H}\cap\mathcal{U}$. Therefore, one has

$$\left(\mathsf{share}_{\mathsf{pk},\mathcal{U},\ell} := \sum_{\mathsf{pk}'\in\mathcal{C}\cap\mathcal{U}\setminus\{\mathsf{pk}\}} a_{\mathsf{pk},\mathcal{U},\ell,\mathsf{pk}'} + \mathsf{share}^0_{\mathsf{pk},\mathcal{H}\cap\mathcal{U},\ell}\right)_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}}$$

are uniformly random shares of $-\sum_{\mathsf{pk}\in\mathcal{C}\cap\mathcal{U}}\mathsf{share}_{\mathsf{pk},\mathcal{U},\ell}$, which is identical to the distribution $\mathcal{R}_{\mathcal{H}\cap\mathcal{U},\ell}$. Since the number of sets $\mathcal{U}$ queried to $\mathsf{QShare}(\cdot,\ell)$ for $\ell \in \mathcal{L}_S \times \mathcal{L}_U$ is bounded q_{QShare}, and each $q_{\mathcal{U}}$ is bounded by $\binom{q_{\mathsf{QNewHon}}}{2}$, one obtains

$$|\mathsf{Adv}_2 - \mathsf{Adv}_3| \leq \frac{1}{2} q_{\mathsf{QNewHon}}(q_{\mathsf{QNewHon}} - 1) q_{\mathsf{QShare}} \cdot \mathsf{Adv}_{\mathsf{DDH}}(\lambda, t + 4q_{\mathsf{QShare}} \times t_{\mathbb{G}}).$$

□

4 Function-Hiding Inner-Product DDFE

In this section, we construct a DDFE scheme for function-hiding inner products from a UZS scheme, a single-input function-hiding IPFE scheme, and an all-or-nothing encapsulation AoNE scheme. The FH-IP-DDFE scheme is proved to be sel-sym-IND secure in the standard model.

4.1 Construction

For every client, let d be an inner-product dimension, let X and Y be a message bound and a function bound of size $\mathsf{poly}(\lambda)$ respectively, let $\mathcal{L}_M$ and $\mathcal{L}_K$ be a message-label space and key-label space respectively. The scheme is described in Figure 10 with the following primitives:

- $\mathsf{IPE} = (\mathsf{iSetup}, \mathsf{iKeyGen}, \mathsf{iEnc}, \mathsf{iDKGen}, \mathsf{iDec})$ be a single-input function-hiding IPFE;
- $\mathsf{UZS} = (\mathsf{SetUp}, \mathsf{KeyGen}, \mathsf{SeedGen}, \mathsf{TokGen}, \mathsf{SeedUpt}, \mathsf{ShareEval})$ be a bilinear-updatable pseudorandom zero-sharing scheme over $\mathbb{G}_1$ for a seeding label space $\mathcal{L}_M$ and an updating label space $\mathcal{L}_K$.
- $\mathsf{AoNE} = (\mathsf{aSetup}, \mathsf{aKeyGen}, \mathsf{aEnc}, \mathsf{aDKGen}, \mathsf{aDec})$ be an all-or-nothing encapsulation scheme.

Correctness. Given $\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow \mathsf{SetUp}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}_{\mathsf{pk}}) \leftarrow \mathsf{KeyGen}()$ $\forall \mathsf{pk} \in \mathcal{PK}$, $\ell_M \in \mathcal{L}_M$, $\ell_K \in \mathcal{L}_K$, $\mathcal{U}_M, \mathcal{U}_K \in \mathcal{S}(\mathcal{PK})$ such that $\mathcal{U}_M = \mathcal{U}_K = \mathcal{U}$, $\boldsymbol{x}_{\mathsf{pk}} \in [-X, X]^d$, $\boldsymbol{y}_{\mathsf{pk}} \in [-Y, Y]^d$ $\forall \mathsf{pk} \in \mathcal{U}$, from the above scheme, one can parse $\mathsf{dk}_{\mathsf{pk}} = (\mathsf{act}_{\mathsf{pk}}, \mathcal{U}, \ell_K)$ and $\mathsf{ct}_{\mathsf{pk}} = (\mathsf{act}'_{\mathsf{pk}}, \mathcal{U}, \ell_M)$ for $\mathsf{pk} \in \mathcal{U}$. By the correctness of the AoNE scheme, one can always recover

$$\mathsf{ict}_{\mathsf{pk}} = \mathsf{iEnc}(\mathsf{isk}_{\mathsf{pk}}, [\boldsymbol{x}_{\mathsf{pk}}, \mathbf{0}^m, \boldsymbol{a}_{\mathsf{pk},\mathcal{U}_M,\ell_M}, 0]_1);$$
$$\mathsf{idk}_{\mathsf{pk}} = \mathsf{iKeyGen}(\mathsf{isk}_{\mathsf{pk}}, [\boldsymbol{y}_{\mathsf{pk}}, \mathbf{0}^m, \boldsymbol{b}_{\mathsf{pk},\mathcal{U}_K,\ell_K}, 0]_2).$$

Construction:

– $\mathsf{SetUp}(1^\lambda)$: It generates $\mathcal{PG} \leftarrow \mathsf{PGGen}(1^\lambda)$ and sets up parameters for the underlying schemes:

$$\mathsf{ipp} \leftarrow \mathsf{iSetup}(1^\lambda) \quad \mathsf{upp} \leftarrow \mathsf{SetUp}(1^\lambda) \quad \mathsf{app} \leftarrow \mathsf{aSetup}(1^\lambda).$$

It returns

$$\mathsf{pp} = (\mathcal{PG}, \mathsf{ipp}, \mathsf{upp}, \mathsf{app}).$$

The parameters pp are implicit to other algorithms.

– $\mathsf{KeyGen}()$: Each client samples

- a PRF key $k_{\mathsf{pk}} \xleftarrow{\$} \mathcal{K}_{\mathsf{PRF}}$;
- UZS keys $(\mathsf{upk}, \mathsf{usk}_{\mathsf{pk}}) \leftarrow \mathsf{KeyGen}()$;
- AoNE keys $(\mathsf{apk}, \mathsf{ask}_{\mathsf{pk}}) \leftarrow \mathsf{aKeyGen}()$.

It returns $\mathsf{pk} = (\mathsf{upk}, \mathsf{apk})$ and $\mathsf{sk}_{\mathsf{pk}} = (k_{\mathsf{pk}}, \mathsf{usk}_{\mathsf{pk}}, \mathsf{ask}_{\mathsf{pk}})$.

– $\mathsf{Enc}(\mathsf{sk}_{\mathsf{pk}}, m)$: It parses $m = (\boldsymbol{x}, \mathcal{U}_M, \ell_M)$ and computes

1. a UZS seed: $[\boldsymbol{a}_{\mathsf{pk},\mathcal{U}_M,\ell_M}]_1 \leftarrow \mathsf{SeedGen}(\mathsf{usk}_{\mathsf{pk}}, (\mathcal{U}_M, \ell_M))$;
2. a random coin for IPE key generation: $\mathsf{coin}_{\mathsf{pk}} \leftarrow \mathsf{PRF}_{k_{\mathsf{pk}}}(\mathcal{U}_M)$;
3. a $2d + |\mathcal{U}_M|$-length IPE secret key: $\mathsf{isk}_{\mathsf{pk}} = \mathsf{iKeyGen}(1^{2d+|\mathcal{U}_M|}; \mathsf{coin}_{\mathsf{pk}})$;
4. an IPE encryption:
$$\mathsf{ict}_{\mathsf{pk}} \leftarrow \mathsf{iEnc}(\mathsf{isk}_{\mathsf{pk}}, [\boldsymbol{x}, \mathbf{0}^d, \boldsymbol{a}_{\mathsf{pk},\mathcal{U}_M,\ell_M}, 0]_1);$$
5. an AoNE layer on $\mathsf{ict}_{\mathsf{pk}}$:
$$\mathsf{act}_{\mathsf{pk}} \leftarrow \mathsf{aEnc}(\mathsf{ask}_{\mathsf{pk}}, (\mathsf{ict}_{\mathsf{pk}}, \mathcal{U}_M, \ell_M, \text{"}dk\text{"}))^a.$$

It returns the ciphertext

$$\mathsf{ct}_{\mathsf{pk}} = (\mathsf{act}_{\mathsf{pk}}, \mathcal{U}_M, \ell_M).$$

If $\mathsf{pk} \notin \mathcal{U}_M$, it returns $\perp$.

– $\mathsf{DKGen}(\mathsf{sk}_{\mathsf{pk}}, k)$: It parses $k = (\boldsymbol{y}, \mathcal{U}_K, \ell_K)$[b] and computes

1. a UZS token: $\boldsymbol{b}_{\mathsf{pk},\mathcal{U}_K,\ell_K} \leftarrow \mathsf{TokGen}(\mathsf{usk}_{\mathsf{pk}}, \mathcal{U}_K, \ell_K)$;
2. a random coin for IPE key generation: $\mathsf{coin}_{\mathsf{pk}} \leftarrow \mathsf{PRF}_{k_{\mathsf{pk}}}(\mathcal{U}_K)$;
3. a $2d + |\mathcal{U}_K|$-length IPE secret key: $\mathsf{isk}_{\mathsf{pk}} = \mathsf{iKeyGen}(1^{2d+|\mathcal{U}_K|}; \mathsf{coin}_{\mathsf{pk}})$;
4. an IPE decryption key:
$$\mathsf{idk}_{\mathsf{pk}} \leftarrow \mathsf{iDKGen}(\mathsf{isk}_{\mathsf{pk}}, [\boldsymbol{y}, \mathbf{0}^d, \boldsymbol{b}_{\mathsf{pk},\mathcal{U}_K,\ell_K}, 0]_2);$$
5. an AoNE layer on $\mathsf{idk}_{\mathsf{pk}}$:
$$\mathsf{act}_{\mathsf{pk}} \leftarrow \mathsf{aEnc}(\mathsf{ask}_{\mathsf{pk}}, (\mathsf{idk}_{\mathsf{pk}}, \mathcal{U}_K, \ell_K, \text{"}dk\text{"})).$$

It returns the decryption key

$$\mathsf{dk}_{\mathsf{pk}} = (\mathsf{act}_{\mathsf{pk}}, \mathcal{U}_K, \ell_K).$$

If $\mathsf{pk} \in \mathcal{U}_K$, it returns $\perp$.

– $\mathsf{Dec}\left((\mathsf{dk}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}_K}, (\mathsf{ct}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}_M}, (\mathcal{U}_M, \ell_M), (\mathcal{U}_K, \ell_K)\right)$: If $\mathcal{U}_M = \mathcal{U}_K = \mathcal{U}$ is not true, it returns $\perp$. Otherwise,

1. it parses $\mathsf{dk}_{\mathsf{pk}} = (\mathsf{act}_{\mathsf{pk}}, \mathcal{U}, \ell_K)$ and recovers the IPE decryption keys
$$(\mathsf{idk}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}} = \mathsf{aDec}((\mathsf{act}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}, \mathcal{U}, \ell_K);$$
2. it parses $\mathsf{ct}_{\mathsf{pk}} = (\mathsf{act}'_{\mathsf{pk}}, \mathcal{U}, \ell_M)$ and recovers the IPE ciphertexts
$$(\mathsf{ict}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}} = \mathsf{aDec}((\mathsf{act}'_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}, \mathcal{U}, \ell_M);$$
3. it computes
$$[\alpha]_T = \sum_{\mathsf{pk}\in\mathcal{U}} \mathsf{iDec}(\mathsf{ict}_{\mathsf{pk}}, \mathsf{idk}_{\mathsf{pk}}).$$

It returns α from $[\alpha]_T$.

[a] AoNE encryptions are additionally randomized by prefixes "ct" and "dk".

[b] Key labels ℓ_K are synchronized for all clients and fresh for each time of decryption key generation.

Fig. 10. DDFE for Function-Hiding Inner Products

The correctness is then implied by the correctness of the IPE scheme and the UZS scheme over $\mathbb{G}_1$:

$$\begin{aligned}
\sum_{\mathsf{pk}\in\mathcal{U}} \mathsf{iDec}(\mathsf{ict}_{\mathsf{pk}}, \mathsf{idk}_{\mathsf{pk}}) &= \sum_{\mathsf{pk}\in\mathcal{U}} [\boldsymbol{x}_{\mathsf{pk}}^\top \cdot \boldsymbol{y}_{\mathsf{pk}} + \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}^\top \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K}]_T \\
&= [\sum_{\mathsf{pk}\in\mathcal{U}} \boldsymbol{x}_{\mathsf{pk}}^\top \cdot \boldsymbol{y}_{\mathsf{pk}}]_T + e\left(\sum_{\mathsf{pk}\in\mathcal{U}} \mathsf{ShareEval}(\mathsf{SeedUpt}(\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_M}, \mathsf{token}_{\mathsf{pk},\mathcal{U},\ell_K})), [1]_2\right) \\
&= [\sum_{\mathsf{pk}\in\mathcal{U}} \boldsymbol{x}_{\mathsf{pk}}^\top \cdot \boldsymbol{y}_{\mathsf{pk}}]_T + e\left(\sum_{\mathsf{pk}\in\mathcal{U}} \mathsf{share}_{\mathsf{pk},\mathcal{U},\ell_M||\ell_K}, [1]_2\right) \\
&= [\sum_{\mathsf{pk}\in\mathcal{U}} \boldsymbol{x}_{\mathsf{pk}}^\top \cdot \boldsymbol{y}_{\mathsf{pk}}]_T + e\left([0]_1, [1]_2\right) \\
&= [\sum_{\mathsf{pk}\in\mathcal{U}} \boldsymbol{x}_{\mathsf{pk}}^\top \cdot \boldsymbol{y}_{\mathsf{pk}}]_T.
\end{aligned}$$

As the inner product $\sum_{\mathsf{pk}\in\mathcal{U}} \boldsymbol{x}_{\mathsf{pk}}^\top \cdot \boldsymbol{y}_{\mathsf{pk}}$ is of size $\mathsf{poly}(\lambda)$, it can always be recovered. □

Remark 6 (Size of Ciphertext/Decryption Key). In the above FH-IP-DDFE construction, if one uses the sel-sym-IND-secure AoNE that is constructed from a rate-1 identity-based encryption and employed in the hybrid-encryption mode with a symmetric encryption as described in [CDSG+20], then the complexity for the size each DDFE ciphertext/decryption key will be $O_\lambda(d + |\mathcal{U}|)$.

4.2 Security Analysis

Theorem 2 (Indistinguishability for FH-IP-DDFE). *If* IPE *is a single-input* sel-sym-fh-IND*-secure FE for function-hiding inner products,* AoNE *is a* sel-sym-nfh-IND*-secure all-or-nothing encapsulation, and* UZS *is an* otu-sta-IND*-secure updatable pseudorandom zero sharing, then the* FH-IP-DDFE *scheme constructed in Figure 10 is* sel-sym-fh-IND *secure (as defined in Definition 6) under the one key-label restriction (as defined in Definition 8) in the standard model.*

Proof. In the selective game, we can fix $\mathcal{PK}$ to be the set of parties generated by $\mathsf{QNewHon}()$ queries, $\mathcal{C}$ to be the set of corrupted parties in $\mathcal{PK}$ and $\mathcal{H} = \mathcal{PK} \setminus \mathcal{C}$ to be the set of honest parties. Let q_{xx} be the number of xx-oracle queries where $\mathsf{xx} \in \{\mathsf{QNewHon}, \mathsf{QEnc}, \mathsf{QDKGen}, \mathsf{QCor}\}$. Given $\lambda \in \mathbb{N}$, we denote by $\mathsf{Adv}_{\mathbf{G}_i}$ the advantage of an PPT adversary $\mathcal{A}$ in each game $\mathbf{G}_i$, and $\mathsf{Adv}_{\mathsf{xx}}$ be the best advantage of any PPT adversary against the primitive xx that is setup with λ. Since the UZS security applies only when there are more than one honest client, we consider two cases: only one honest client and more than one honest client.

The case of one honest client $\mathcal{H} = \{\mathsf{pk}^\star\}$. By facts in Remark 2, given any $\mathcal{U} \in \mathcal{S}(\mathcal{PK})$, for any $\mathsf{QEnc}(\mathsf{pk}^\star, \boldsymbol{x}^0, \boldsymbol{x}^1, \mathcal{U}, \ell_M)$[6] query and for any $\mathsf{QDKGen}(\mathsf{pk}^\star, \boldsymbol{y}^0, \boldsymbol{y}^1, \mathcal{U}, \ell_K)$, then it must hold that

$$\boldsymbol{x}^{0\top}\boldsymbol{y}^0 - \boldsymbol{x}^{1\top}\boldsymbol{y}^1 = 0.$$

Let κ be a bijective map from the set

$$\{\mathcal{U} \in \mathcal{S}(\mathcal{PK}) : \mathsf{QEnc}(\mathsf{pk}^\star, \cdot, \cdot, \mathcal{U}, \cdot) \text{ or } \mathsf{QDKGen}(\mathsf{pk}^\star, \cdot, \cdot, \mathcal{U}, \cdot) \text{ was sent}\}$$

to $[q_u]$ for some integer q_u. With $j \in [q_u]$, we proceed by the following sequence of hybrid games:

$\mathbf{G}_0^{\mathsf{pk}^\star}$: This is the real game with one honest client $\mathsf{pk}^\star$.

$\mathbf{G}_1^{\mathsf{pk}^\star}$: The change is that the pseudorandom function $\mathsf{PRF}_{k_{\mathsf{pk}^\star}}$ is replaced by a random function RF. The indistinguishability is implied by the security of the PRF:

$$\left|\mathsf{Adv}_{\mathbf{G}_0^{\mathsf{pk}^\star}} - \mathsf{Adv}_{\mathbf{G}_1^{\mathsf{pk}^\star}}\right| \leq \mathsf{Adv}_{\mathsf{PRF}}.$$

[6] Without losing the formality, we omit subscript indexes in clear contexts and use the format $(\cdot, \cdot, \cdot, \mathcal{U}, \ell)$ instead of $(\cdot, (\cdot, \mathcal{U}, \ell), (\cdot, \mathcal{U}, \ell))$ for encryption/decryption-key queries.

$\mathbf{G}_{1.j}^{\mathsf{pk}^\star}$: The change is that instead of depending on the bit b, for every $\mathsf{QEnc}(\mathsf{pk}, \boldsymbol{x}^0, \boldsymbol{x}^1, \mathcal{U}, \ell_M)$ query and for every $\mathsf{QDKGen}(\mathsf{pk}, \boldsymbol{y}^0, \boldsymbol{y}^1, \mathcal{U}, \ell_K)$ query where $\kappa(\mathcal{U}) = j$, the challenger chooses $\boldsymbol{x}^0$ and $\boldsymbol{y}^0$ to generate the answers respectively.

We note that in the symmetric-key variant of the security game, there is no information on the challenge bit b from the QEnc and QDKGen queries on corrupted clients. Then we can assume that $\mathbf{G}_{1.0}^{\mathsf{pk}^\star}$ corresponds to $\mathbf{G}_1^{\mathsf{pk}^\star}$ and $\mathbf{G}_{1.q_u}^{\mathsf{pk}^\star}$ corresponds to the game where the adversarial advantage is 0. To transition between $\mathbf{G}_{1.j-1}^{\mathsf{pk}^\star}$ and $\mathbf{G}_{1.j}^{\mathsf{pk}^\star}$, we construct an adversary $\mathcal{B}$ against the IPE security as follows:

- For every $\mathsf{QEnc}(\mathsf{pk}^\star, \boldsymbol{x}^0, \boldsymbol{x}^1, \mathcal{U}, \ell_M)$ query and every $\mathsf{QDKGen}(\mathsf{pk}^\star, \boldsymbol{y}^0, \boldsymbol{y}^1, \mathcal{U}, \ell_K)$ query where $\kappa(\mathcal{U}) = j$ from $\mathcal{A}$, then $\mathcal{B}$ queries the IPE challenge oracles with the following messages and functions respectively

$$[\boldsymbol{m}^\gamma]_1 = [\boldsymbol{x}^\gamma, \mathbf{0}^d, \boldsymbol{a}_{\mathsf{pk}^\star,\mathcal{U},\ell_M}, 0]_1,$$
$$[\boldsymbol{k}^\gamma]_2 = [\boldsymbol{y}^\gamma, \mathbf{0}^d, \boldsymbol{b}_{\mathsf{pk}^\star,\mathcal{U},\ell_K}, 0]_2$$

  where $\gamma \in \{0,1\}$ and $\boldsymbol{a}_{\mathsf{pk}^\star,\mathcal{U},\ell_M}$, $\boldsymbol{b}_{\mathsf{pk}^\star,\mathcal{U},\ell_K}$ are generated from the UZS scheme. Then $\mathcal{B}$ receives the answers from IPE oracles to complete the reply to $\mathcal{A}$.
- $\mathcal{B}$ outputs $\mathcal{A}$'s guess for the challenge bit IPE.b.

The admissibility condition holds for IPE as ${\boldsymbol{x}^0}^\top \boldsymbol{y}^0 - {\boldsymbol{x}^1}^\top \boldsymbol{y}^1 = 0$ and $(\boldsymbol{a}_{\mathsf{pk}^\star,\mathcal{U},\ell_M}, \boldsymbol{b}_{\mathsf{pk}^\star,\mathcal{U},\ell_K})$ are independent of IPE.b. Therefore, we have $\left|\mathsf{Adv}_{\mathbf{G}_{1.j-1}^{\mathsf{pk}^\star}} - \mathsf{Adv}_{\mathbf{G}_{1.j}^{\mathsf{pk}^\star}}\right| \leq \mathsf{Adv}_{\mathsf{IPE}}$ and then

$$\mathsf{Adv}_{\mathbf{G}_0^{\mathsf{pk}^\star}} \leq (q_{\mathsf{QEnc}} + q_{\mathsf{QDKGen}}) \cdot \mathsf{Adv}_{\mathsf{IPE}} + \mathsf{Adv}_{\mathsf{PRF}}.$$

The case of more than one honest client. Let $\mathcal{Q}_M$ and $\mathcal{Q}_K$[7] be the set of encryption queries and decryption key queries sent in one shot by $\mathcal{A}$ respectively. We proceed via a hybrid argument: for readability, we describe the global changes in the IND game by using the games $\mathbf{G}_0$, $\mathbf{G}_1$, $\mathbf{G}_2$ and $\mathbf{G}_3$ (see Figure 11); the transition between $\mathbf{G}_2$ and $\mathbf{G}_3$ requires intermediate games $\mathbf{G}_{2.1}$ and $\mathbf{G}_{2.1.\ell_K}$ for each queried key label $\ell_K \in \mathcal{Q}_K$ (see Figure 6), and the transition between $\{\mathbf{G}_{2.1.\ell_K}\}_{\ell_K \in \mathcal{Q}_K}$ requires intermediate $(\mathbf{G}^\star_{\ell_K.i})_{i\in[4]}$ (see Figure 7) for each ℓ_K. Notably, the game $\mathbf{G}_0$ corresponds to sel-sym-fh-IND security game as defined in Definition 6, and the game $\mathbf{G}_3$ corresponds to the case where adversary's advantage is 0 since there is no challenge bit b.

Game $\mathbf{G}_1$: The change is that the challenger uses a random function $\mathsf{RF}_{\mathsf{pk}}$ instead of $\mathsf{PRF}_{k_{\mathsf{pk}}}$. The indistinguishability is implied by the security of the pseudorandom function, given in Lemma 4.

Game $\mathbf{G}_2$: When $\mathsf{pk} \in \mathcal{H}$, a decryption key query $(\mathsf{pk}, \boldsymbol{y}^0, \boldsymbol{y}^1, \mathcal{U}_K, \ell_K) \in \mathcal{Q}_K$ is said to be incomplete if there exists $\mathsf{pk}' \in \mathcal{H} \cap \mathcal{U}_K$ and the decryption key query $(\mathsf{pk}', \boldsymbol{y'}^0, \boldsymbol{y'}^1, \mathcal{U}_K, \ell_K) \notin \mathcal{Q}_K$. For that query, $\mathsf{act}_{\mathsf{pk}}$ is changed to the encapsulation of $(0, \mathcal{U}_K, \ell_K, \text{"}dk\text{"})$. Similarly, when $\mathsf{pk} \in \mathcal{H}$, an encryption query $(\mathsf{pk}, \boldsymbol{x}^0, \boldsymbol{x}^1, \mathcal{U}_M, \ell_M) \in \mathcal{Q}_M$ is said to be incomplete if there exists $\mathsf{pk}'\mathsf{pk}' \in \mathcal{H} \cap \mathcal{U}_M$ and the encryption query $(\mathsf{pk}', \boldsymbol{x'}^0, \boldsymbol{x'}^1, \mathcal{U}_M, \ell_M) \notin \mathcal{Q}_M$. For that query, $\mathsf{act}_{\mathsf{pk}}$ is changed to the encapsulation of $(0, \mathcal{U}_M, \ell_M, \text{"}ct\text{"})$. The indistinguishability is implied by the security of the AoNE scheme, given in Lemma 5.

Game $\mathbf{G}_{2.1}$: The change is that for every complete encryption query on $(\mathsf{pk}, \boldsymbol{x}^0, \boldsymbol{x}^1, \mathcal{U}_M, \ell_M)$, the challenger sets the IPE message as $(\boldsymbol{x}^b, \boldsymbol{x}^0, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}, 0)$. The indistinguishability is implied by the security of the IPE scheme, given in Lemma 6.

Assume that the set of queried key labels is ordered, for every $\ell_K \in \mathcal{Q}_K$, one has the following intermediate games:

Game $\mathbf{G}^\star_{\ell_K.1}$: The change is that for every complete decryption key query on $(\mathsf{pk}, \boldsymbol{y}^0, \boldsymbol{y}^1, \mathcal{U}, \ell'_K)$ for $\ell'_K = \ell_K$, the challenger sets the IPE key as $(\mathbf{0}^d, \mathbf{0}^d, \mathbf{0}^{|\mathcal{U}|-1}, 1)$, and for every complete encryption query on $(\mathsf{pk}, \boldsymbol{x}^0, \boldsymbol{x}^1, \mathcal{U}, \ell_M)$, the challenger sets the IPE message as $(\boldsymbol{x}^b, \boldsymbol{x}^0, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}, \boldsymbol{a}^\top_{\mathsf{pk},\mathcal{U},\ell_M} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K} + {\boldsymbol{x}^b}^\top \cdot \boldsymbol{y}^b)$. The indistinguishability is implied by the security of the IPE scheme, given in Lemma 7.

[7] $\mathcal{Q}_M$ and $\mathcal{Q}_K$ contain elements of the form $(\mathsf{pk}, \cdot, \cdot, \mathcal{U}, \ell)$. For a string $\mathrm{xx} \in \{0,1\}^*$, we denote by $\mathrm{xx} \in \mathcal{Q}_M$ or $\mathrm{xx} \in \mathcal{Q}_K$ if there exists a query containing xx.

Game $\mathbf{G}^{\star}_{\ell_K.2}$: The change is that for every complete encryption query on $(\mathsf{pk}, \boldsymbol{x}^0, \boldsymbol{x}^1, \mathcal{U}, \ell_M)$, the challenger sets the IPE message as $(\boldsymbol{x}^b, \boldsymbol{x}^0, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}, R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K} + \boldsymbol{x}^{b\top} \cdot \boldsymbol{y}^b)$ where $(R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K})_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}}$ are sampled uniformly such that

$$\sum_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}} R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K} = \sum_{\mathsf{pk}\in\mathcal{C}\cap\mathcal{U}} \boldsymbol{a}^{\top}_{\mathsf{pk},\mathcal{U},\ell_M} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K}.$$

The indistinguishability is implied by the security of the UZS scheme, given in Lemma 8.

Game $\mathbf{G}^{\star}_{\ell_K.3}$: The change is that for every complete encryption query on $(\mathsf{pk}, \boldsymbol{x}^0, \boldsymbol{x}^1, \mathcal{U}, \ell_M)$, the challenger sets the IPE message as $(\boldsymbol{x}^b, \boldsymbol{x}^0, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}, R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K} + \boldsymbol{x}^{0\top} \cdot \boldsymbol{y}^0)$. The indistinguishability is perfect, given in Lemma 9.

Game $\mathbf{G}^{\star}_{\ell_K.4}$: The change is that for every complete encryption query on $(\mathsf{pk}, \boldsymbol{x}^0, \boldsymbol{x}^1, \mathcal{U}, \ell_M)$, the challenger sets the IPE message as $(\boldsymbol{x}^b, \boldsymbol{x}^0, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}, \boldsymbol{a}^{\top}_{\mathsf{pk},\mathcal{U},\ell_M} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K} + \boldsymbol{x}^{0\top} \cdot \boldsymbol{y}^0)$. This change is symmetric to the change in $\mathbf{G}^{\star}_{\ell_K.2}$ and then the indistinguishability is implied by the security of the UZS scheme.

Game $\mathbf{G}_{2.1.(\ell_K+1)}$: For the subsequent label $(\ell_K + 1)$ of ℓ_K in $\mathcal{Q}_K$, the change is that for every complete decryption key query on $(\mathsf{pk}, \boldsymbol{y}^0, \boldsymbol{y}^1, \mathcal{U}, \ell_K)$, the challenger sets the IPE key as $(\mathbf{0}^d, \boldsymbol{y}^0_{\mathsf{pk}}, \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K}, 0)$, and for every complete encryption query on $(\mathsf{pk}, \boldsymbol{x}^0, \boldsymbol{x}^1, \mathcal{U}, \ell_M)$, the challenger sets the IPE message as $(\boldsymbol{x}^b, \boldsymbol{x}^0, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}, 0)$. This change is symmetric to the change in $\mathbf{G}^{\star}_{\ell_K.1}$ and then the indistinguishability is implied by the security of the IPE scheme.

By using a recursive transition through all $\ell_K \in \mathcal{Q}_K$, one comes to the final game:

Game $\mathbf{G}_3$: In this game, to answer any complete encryption query and any complete decryption key generation query, the challenger sets the IPE key as $(\mathbf{0}^d, \boldsymbol{y}^0, \boldsymbol{b}, 0)$ and the IPE message as $(\mathbf{0}^d, \boldsymbol{x}^0, \boldsymbol{a}, 0)$ respectively. There is thus no dependence on the challenge bit b in this game, so $\mathsf{Adv}_{\mathbf{G}_3} = 0$.

From the transitions above, one completes the theorem by having

$$\begin{aligned}\mathsf{Adv}^{\text{sel-sym-fh}}_{\mathsf{FH\text{-}IP\text{-}DDFE}} \leq\ & [2q_{\mathsf{QDKGen}}(q_{\mathsf{QEnc}} + q_{\mathsf{QDKGen}}) + q_{\mathsf{QEnc}}] \cdot \mathsf{Adv}^{\text{sel-sym-fh}}_{\mathsf{IPE}} + 2q_{\mathsf{QDKGen}} \cdot \mathsf{Adv}^{\text{otu-sta}}_{\mathsf{UZS}} \\ & + \mathsf{Adv}^{\text{sel-sym-nfh}}_{\mathsf{AoNE}} + (q_{\mathsf{QNewHon}} - q_{\mathsf{QCor}}) \cdot \mathsf{Adv}_{\mathsf{PRF}}.\end{aligned}$$

□

Lemma 4 (FH-IP-DDFE: Transition from $\mathbf{G}_0$ to $\mathbf{G}_1$). *For any PPT adversary $\mathcal{A}$, the advantage in distinguishing two games is*

$$|\mathsf{Adv}_{\mathbf{G}_0} - \mathsf{Adv}_{\mathbf{G}_1}| \leq (q_{\mathsf{QNewHon}} - q_{\mathsf{QCor}}) \cdot \mathsf{Adv}_{\mathsf{PRF}}.$$

Proof. We proceed by using multiple hybrid games for each $\mathsf{pk} \in \mathcal{H}$. For each transition, we build an adversary $\mathcal{B}$ against the IND security of PRF from an adversary $\mathcal{A}$ that distinguishes two games in the transition. To simulate a FH-IP-DDFE challenger, $\mathcal{B}$ uses the PRF oracle, instead of generating by itself the PRF key k_{pk} to handle all $\mathsf{PRF}_{k_{\mathsf{pk}}}$ related operations, and finally outputs $\mathcal{A}$'s guess for the challenge bit PRF.b. Since the number of honest parties is bounded by $(q_{\mathsf{QNewHon}} - q_{\mathsf{QCor}})$, one completes the proof. □

Lemma 5 (FH-IP-DDFE: Transition from $\mathbf{G}_1$ to $\mathbf{G}_2$). *For any PPT adversary $\mathcal{A}$, the advantage in distinguishing two games is*

$$|\mathsf{Adv}_{\mathbf{G}_1} - \mathsf{Adv}_{\mathbf{G}_2}| \leq \mathsf{Adv}^{\text{sel-sym-nfh}}_{\mathsf{AoNE}}.$$

Proof. We build an adversary $\mathcal{B}$ against the sel-sym-nfh-IND security game of AoNE from an adversary $\mathcal{A}$ that distinguishes between $\mathbf{G}_1$ and $\mathbf{G}_2$. To simulate a FH-IP-DDFE challenger, $\mathcal{B}$ uses the oracles of the AoNE challenger to handle all AoNE related operations.

– For each $\mathsf{QDKGen}(\mathsf{pk}, \boldsymbol{y}^0, \boldsymbol{y}^1, \mathcal{U}_K, \ell_K)$ query from $\mathcal{A}$, the adversary $\mathcal{B}$ prepares the IPE key $\mathsf{idk}_{\mathsf{pk}}$. If the query is complete, $\mathcal{B}$ sends $(\mathsf{pk}, \mathsf{idk}_{\mathsf{pk}}, \mathsf{idk}_{\mathsf{pk}}, \mathcal{U}_K, \ell_K, \text{"}dk\text{"})$ to the AoNE encryption oracle. Otherwise, it sends $(\mathsf{pk}, \mathsf{idk}_{\mathsf{pk}}, 0, \mathcal{U}_K, \ell_K, \text{"}dk\text{"})$. Upon receiving the oracle's reply $\mathsf{act}_{\mathsf{pk}}$, $\mathcal{B}$ uses this ciphertext to complete the reply to QDKGen query from $\mathcal{A}$.

$\mathbf{G}_0$, $\mathbf{G}_1$, $\mathbf{G}_2$, $\mathbf{G}_3$:

$\mathsf{pp} \leftarrow \mathsf{SetUp}(1^\lambda)$, $b \xleftarrow{\$} \{0,1\}$
$(\mathcal{PK}, \mathsf{st}_1) \leftarrow \mathcal{A}(\mathsf{pp})$
$(\mathsf{pk})_{\mathsf{pk}\in\mathcal{PK}} \leftarrow \mathsf{QNewHon}([\mathcal{PK}])$
$(\mathcal{Q}_M, \mathcal{Q}_K, \mathcal{C}, \mathsf{st}_2) \leftarrow \mathcal{A}((\mathsf{pk})_{\mathsf{pk}\in\mathcal{PK}}, \mathsf{st}_1)$
$\mathsf{st}_{\mathcal{Q}_M} \leftarrow \mathsf{QEnc}([\mathcal{Q}_M])$, $\mathsf{st}_{\mathcal{Q}_K} \leftarrow \mathsf{QDKGen}([\mathcal{Q}_K])$, $\mathsf{st}_{\mathcal{C}} \leftarrow \mathsf{QCor}([\mathcal{C}])$
$b' \leftarrow \mathcal{A}(\mathsf{st}_{\mathcal{Q}_M}, \mathsf{st}_{\mathcal{Q}_K}, \mathsf{st}_{\mathcal{C}}, \mathsf{st}_2)$
Output: b' if Condition $(*)$ is satisfied, or $b' \leftarrow \{0,1\}$ otherwise.

$\underline{\mathsf{QNewHon}()}$:
$k_{\mathsf{pk}} \leftarrow \mathcal{K}_{\mathsf{PRF}}$; $(\mathsf{upk}, \mathsf{usk}_{\mathsf{pk}}) \leftarrow \mathsf{KeyGen}()$; $(\mathsf{apk}, \mathsf{ask}_{\mathsf{pk}}) \leftarrow \mathsf{aKeyGen}()$
$\mathsf{pk} = (\mathsf{upk}, \mathsf{apk})$; $\mathsf{sk}_{\mathsf{pk}} = (k_{\mathsf{pk}}, \mathsf{usk}_{\mathsf{pk}}, \mathsf{ask}_{\mathsf{pk}})$
Store $(\mathsf{pk}, \mathsf{sk}_{\mathsf{pk}})$ and return pk.

$\underline{\mathsf{QDKGen}(\mathsf{pk}, \boldsymbol{y}^0, \boldsymbol{y}^1, (\mathcal{U}, \ell))}$:
$\boldsymbol{b} \leftarrow \mathsf{TokGen}(\mathsf{usk}_{\mathsf{pk}}, (\mathcal{U}, \ell))$
$\mathsf{coin} \leftarrow \mathsf{PRF}_{k_{\mathsf{pk}}}(\mathcal{U})$ $\mathsf{coin} \leftarrow \mathsf{RF}_{\mathsf{pk}}(\mathcal{U})\ \forall \mathsf{pk} \in \mathcal{H}$
$\mathsf{isk} \leftarrow \mathsf{iKeyGen}(1^{2d+\mathcal{U}}; \mathsf{coin})$
$\boldsymbol{k} = (\boldsymbol{y}^b, \mathbf{0}^d, \boldsymbol{b}, 0)$ $\boldsymbol{k} = (\mathbf{0}^d, \boldsymbol{y}^0, \boldsymbol{b}, 0)\ \forall \mathsf{pk} \in \mathcal{H}$
$\mathsf{idk} \leftarrow \mathsf{iDKGen}(\mathsf{isk}, [\boldsymbol{k}]_2)$
If $\exists \mathsf{pk}' \in \mathcal{H}$ such that $(\mathsf{pk}', \boldsymbol{y'}^0, \boldsymbol{y'}^1, \mathcal{U}, \ell) \notin \mathcal{Q}_K$:
$\mathsf{act} \leftarrow \mathsf{aEnc}(\mathsf{ask}_{\mathsf{pk}}, (0, \mathcal{U}, \ell))$
$\mathsf{act} \leftarrow \mathsf{aEnc}(\mathsf{ask}_{\mathsf{pk}}, (\mathsf{idk}, \mathcal{U}, \ell, "dk"))$
Return $(\mathsf{act}, \mathcal{U}, \ell)$.

$\underline{\mathsf{QEnc}(\mathsf{pk}, \boldsymbol{x}^0, \boldsymbol{x}^1, (\mathcal{U}, \ell))}$:
$[\boldsymbol{a}]_1 \leftarrow \mathsf{SeedGen}(\mathsf{usk}_{\mathsf{pk}}, (\mathcal{U}, \ell))$
$\mathsf{coin} \leftarrow \mathsf{PRF}_{k_{\mathsf{pk}}}(\mathcal{U})$ $\mathsf{coin} \leftarrow \mathsf{RF}_{\mathsf{pk}}(\mathcal{U})\ \forall \mathsf{pk} \in \mathcal{H}$
$\mathsf{isk} \leftarrow \mathsf{iKeyGen}(1^{2d+\mathcal{U}}; \mathsf{coin})$
$\boldsymbol{m} = (\boldsymbol{x}^b, \mathbf{0}^d, \boldsymbol{a}, 0)$ $\boldsymbol{m} = (\mathbf{0}^d, \boldsymbol{x}^0, \boldsymbol{a}, 0)\ \forall \mathsf{pk} \in \mathcal{H}$
$\mathsf{ict} \leftarrow \mathsf{iEnc}(\mathsf{isk}, [\boldsymbol{m}]_1)$
If $\exists \mathsf{pk}' \in \mathcal{H}$ such that $(\mathsf{pk}', \boldsymbol{x'}^0, \boldsymbol{x'}^1, \mathcal{U}, \ell) \notin \mathcal{Q}_M$:
$\mathsf{act} \leftarrow \mathsf{aEnc}(\mathsf{ask}_{\mathsf{pk}}, (0, \mathcal{U}, \ell))$
$\mathsf{act} \leftarrow \mathsf{aEnc}(\mathsf{ask}_{\mathsf{pk}}, (\mathsf{ict}_{\mathsf{pk}}, \mathcal{U}, \ell, "ct"))$
Return $(\mathsf{act}, \mathcal{U}, \ell)$.

$\underline{\mathsf{QCor}(\mathsf{pk})}$:
If $\mathsf{pk} \notin \mathcal{PK}$, return $\perp$.
Return $\mathsf{sk}_{\mathsf{pk}} = (k_{\mathsf{pk}}, \mathsf{usk}_{\mathsf{pk}}, \mathsf{ask}_{\mathsf{pk}})$.

Fig. 11. Games for the sel-sym-fh-IND proof of FH-IP-DDFE in Theorem 2

- For each $\mathsf{QEnc}(\mathsf{pk}, \boldsymbol{x}^0, \boldsymbol{x}^1, \mathcal{U}_M, \ell_M)$ query from $\mathcal{A}$, the adversary $\mathcal{B}$ prepares the IPE ciphertext $\mathsf{ict}_{\mathsf{pk}}$. If the query is complete, $\mathcal{B}$ sends $(\mathsf{pk}, \mathsf{ict}_{\mathsf{pk}}, \mathsf{ict}_{\mathsf{pk}}, \mathcal{U}_M, \ell_M, "ct")$ to the AoNE encryption oracle. Otherwise, it sends $(\mathsf{pk}, \mathsf{ict}_{\mathsf{pk}}, 0, \mathcal{U}_M, \ell_M, "ct")$. Upon receiving the oracle's reply $\mathsf{act}_{\mathsf{pk}}$, $\mathcal{B}$ uses this ciphertext to complete the reply to QEnc query from $\mathcal{A}$.
- $\mathcal{B}$ outputs $\mathcal{A}$'s guess for the challenge bit $\mathsf{AoNE}.b$.

The admissibility condition $(*)$ of AoNE holds since each pair of challenge messages differ only when an incomplete query was made. We also use prefix "ct" and "dk" to prevent possible combinations between key queries and ciphertext queries in decryption. Therefore, when $\mathsf{AoNE}.b = 0$, $\mathcal{B}$ is playing $\mathbf{G}_1$; when $\mathsf{AoNE}.b = 1$, $\mathcal{B}$ is playing $\mathbf{G}_2$. □

Lemma 6 (FH-IP-DDFE: Transition from $\mathbf{G}_2$ to $\mathbf{G}_{2.1}$). *For any PPT adversary $\mathcal{A}$, the advantage in distinguishing two games is*

$$|\mathsf{Adv}_{\mathbf{G}_2} - \mathsf{Adv}_{\mathbf{G}_{2.1}}| \leq q_{\mathsf{QEnc}} \cdot \mathsf{Adv}^{\text{sel-sym-fh}}_{\mathsf{IPE}}.$$

Proof. We proceed by using multiple hybrid games for each pair of honest $(\mathsf{pk}, \mathcal{U}) \in \mathcal{Q}_M$. For each transition, we build an adversary $\mathcal{B}$ against the sel-sym-fh-IND security of IPE from an adversary $\mathcal{A}$ that distinguishes between two games in the transition. To simulate a FH-IP-DDFE challenger, $\mathcal{B}$ uses the IPE oracles to handle all IPE related operations for the reply of each $(\mathsf{pk}, \mathcal{U})$-involved query from $\mathcal{A}$.

- For each complete $\mathsf{QDKGen}(\mathsf{pk}, \boldsymbol{y}^0, \boldsymbol{y}^1, \mathcal{U}, \ell_K)$ query, $\mathcal{B}$ prepares the IPE key as
$$\boldsymbol{k}^0 = \boldsymbol{k}^1 = (\boldsymbol{y}^b, \mathbf{0}^d, \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K}, 0)$$
and sends $(\mathsf{pk}, [\boldsymbol{k}^0]_2, [\boldsymbol{k}^1]_2)$ to the IPE decryption key oracle. It uses the returned decryption key $\mathsf{idk}_{\mathsf{pk}}$ to complete the reply to $\mathcal{A}$.
- For each complete $\mathsf{QEnc}(\mathsf{pk}, \boldsymbol{x}^0, \boldsymbol{x}^1, \mathcal{U}, \ell_M)$ query, $\mathcal{B}$ prepares the IPE message as
$$\boldsymbol{m}^0 = (\boldsymbol{x}^b, \mathbf{0}^d, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}, 0);$$
$$\boldsymbol{m}^1 = (\boldsymbol{x}^b, \boldsymbol{x}^0, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}, 0);$$
and sends $(\mathsf{pk}, [\boldsymbol{m}^0]_1, [\boldsymbol{m}^1]_1)$ to the IPE encryption oracle. It uses the returned ciphertext $\mathsf{ict}_{\mathsf{pk}}$ to complete the reply to $\mathcal{A}$.
- $\mathcal{B}$ outputs $\mathcal{A}$'s guess for the challenge bit $\mathsf{IPE}.b$.

The admissibility condition $(*)$ of IPE in each transition holds since one always has ${\boldsymbol{k}^0}^\top \cdot \boldsymbol{m}^0 = {\boldsymbol{k}^1}^\top \cdot \boldsymbol{m}^0$. Therefore, in each transition of the multiple hybrid games for each $(\mathsf{pk}, \mathcal{U})$, when $\mathsf{IPE}.b = 0$, $\mathcal{A}$ is playing the previous game; when $\mathsf{IPE}.b = 1$, $\mathcal{A}$ is playing the subsequent game. As the number of pairs $(\mathsf{pk}, \mathcal{U}) \in \mathcal{Q}_M$ is bounded by q_{QEnc}, one completes the proof by having

$$|\mathsf{Adv}_{\mathbf{G}_2} - \mathsf{Adv}_{\mathbf{G}_{2.1}}| \leq q_{\mathsf{QEnc}} \cdot \mathsf{Adv}^{\text{sel-sym-fh}}_{\mathsf{IPE}}.$$

Lemma 7 (FH-IP-DDFE: Transition from $\mathbf{G}_{2.1.\ell_K}$ to $\mathbf{G}^*_{\ell_K.1}$). *For any PPT adversary $\mathcal{A}$, the advantage in distinguishing two games is*

$$\left|\mathsf{Adv}_{\mathbf{G}_{2.1.\ell_K}} - \mathsf{Adv}_{\mathbf{G}^*_{\ell_K.1}}\right| \leq (q_{\mathsf{QEnc}} + q_{\mathsf{QDKGen}}) \cdot \mathsf{Adv}^{\text{sel-sym-fh}}_{\mathsf{IPE}}.$$

Proof. We proceed by using multiple hybrid games for each pair of honest $(\mathsf{pk}, \mathcal{U}) \in \mathcal{Q}_M \cup \mathcal{Q}_K$. We build an adversary $\mathcal{B}$ against the sel-sym-fh-IND security of IPE from an adversary $\mathcal{A}$ that distinguishes between two games in the transition. To simulate a FH-IP-DDFE challenger, $\mathcal{B}$ uses the IPE oracles to handle all IPE related operations for the reply of each $(\mathsf{pk}, \mathcal{U})$-involved query from $\mathcal{A}$.

- For each complete $\mathsf{QDKGen}(\mathsf{pk}, \boldsymbol{y}^0_{\mathsf{pk},\ell'_K}, \boldsymbol{y}^1_{\mathsf{pk},\ell'_K}, \mathcal{U}, \ell'_K)$[8] query, $\mathcal{B}$ prepares the IPE key as

If $\ell'_K < \ell_K$: $\boldsymbol{k}^0 = \boldsymbol{k}^1 = (\mathbf{0}^d, \boldsymbol{y}^0_{\mathsf{pk},\ell'_K}, \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell'_K}, 0)$

If $\ell'_K = \ell_K$:

$\boldsymbol{k}^0 = (\boldsymbol{y}^b_{\mathsf{pk},\ell_K}, \mathbf{0}^d, \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K}, 0)$

$\boldsymbol{k}^1 = (\mathbf{0}^d, \mathbf{0}^d, \mathbf{0}^{|\mathcal{U}|-1}, 1)$

If $\ell'_K > \ell_K$: $\boldsymbol{k}^0 = \boldsymbol{k}^1 = (\boldsymbol{y}^b_{\mathsf{pk},\ell'_K}, \mathbf{0}^d, \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell'_K}, 0)$

[8] We additionally use subscripts ℓ'_K and ℓ_K for $\boldsymbol{y}^b_{\mathsf{pk}}$ to differentiate the vector $\boldsymbol{y}^b_{\mathsf{pk}}$ that generates the decryption key under ℓ_K from those of other key labels.

and sends $(\mathsf{pk}, [\boldsymbol{k}^0]_2, [\boldsymbol{k}^1]_2)$ to the IPE decryption key generation oracle. It uses the returned decryption key $\mathsf{idk}_{\mathsf{pk}}$ to complete the reply to $\mathcal{A}$.
– For each complete $\mathsf{QEnc}(\mathsf{pk}, \boldsymbol{x}^0_{\mathsf{pk}}, \boldsymbol{x}^1_{\mathsf{pk}}, \mathcal{U}, \ell_M)$ query, $\mathcal{B}$ prepares the IPE message as

$$\boldsymbol{m}^0 = (\boldsymbol{x}^b_{\mathsf{pk}}, \boldsymbol{x}^0_{\mathsf{pk}}, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}, 0);$$
$$\boldsymbol{m}^1 = (\boldsymbol{x}^b_{\mathsf{pk}}, \boldsymbol{x}^0_{\mathsf{pk}}, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}, \boldsymbol{a}^\top_{\mathsf{pk},\mathcal{U},\ell_M} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K} + {\boldsymbol{x}^b_{\mathsf{pk}}}^\top \cdot \boldsymbol{y}^b_{\mathsf{pk},\ell_K});$$

and sends $(\mathsf{pk}, [\boldsymbol{m}^0]_1, [\boldsymbol{m}^1]_1)$ to the IPE encryption oracle. It uses the returned decryption key $\mathsf{ict}_{\mathsf{pk}}$ to complete the reply to $\mathcal{A}$.
– $\mathcal{B}$ outputs $\mathcal{A}$'s guess for the challenge bit IPE.b.

The admissibility condition $(*)$ of IPE in each transition holds since one always has ${\boldsymbol{k}^0}^\top \cdot \boldsymbol{m}^0 = {\boldsymbol{k}^1}^\top \cdot \boldsymbol{m}^0$. Therefore, in each transition of the multiple hybrid games for each $(\mathsf{pk},\mathcal{U})$, when IPE.$b = 0$, $\mathcal{A}$ is playing the previous game; when IPE.$b = 1$, $\mathcal{A}$ is playing the subsequent game. Since the number of pairs $(\mathsf{pk},\mathcal{U}) \in \mathcal{Q}_M \cup \mathcal{Q}_K$ is bounded by $(q_{\mathsf{QEnc}} + q_{\mathsf{QDKGen}})$, one has

$$\left|\mathsf{Adv}_{\mathbf{G}_{2.1.\ell_K}} - \mathsf{Adv}_{\mathbf{G}^*_{\ell_K.1}}\right| \leq (q_{\mathsf{QEnc}} + q_{\mathsf{QDKGen}}) \cdot \mathsf{Adv}^{\text{sel-sym-fh}}_{\mathsf{IPE}}.$$

□

Lemma 8 (FH-IP-DDFE: Transition from $\mathbf{G}^*_{\ell_K.1}$ to $\mathbf{G}^*_{\ell_K.2}$). *For any PPT adversary $\mathcal{A}$, the advantage in distinguishing two games is*

$$\left|\mathsf{Adv}_{\mathbf{G}^*_{\ell_K.1}} - \mathsf{Adv}_{\mathbf{G}^*_{\ell_K.2}}\right| \leq \mathsf{Adv}^{\text{otu-sta}}_{\mathsf{UZS}}.$$

Proof. We build an adversary $\mathcal{B}$ against the otu-sta-IND security of UZS from an adversary $\mathcal{A}$ that distinguishes between two games in the transition. To simulate a FH-IP-DDFE challenger, $\mathcal{B}$ uses the UZS oracles to handle all UZS related operations.

– For each complete $\mathsf{QDKGen}(\mathsf{pk}, \boldsymbol{y}^0_{\mathsf{pk}}, \boldsymbol{y}^1_{\mathsf{pk}}, \mathcal{U}, \ell'_K)$ query, the adversary $\mathcal{B}$ prepares the IPE key as
  - If $\ell'_K \neq \ell_K$: it obtains $\boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell'_K} \leftarrow \mathsf{QTokGen}(\mathsf{pk},\mathcal{U},\ell'_K)$ to complete the key $\boldsymbol{k}$.
  - If $\ell'_K = \ell_K$: it does not have to obtain $\boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K}$ as $\boldsymbol{k} = (\mathbf{0}^d, \mathbf{0}^d, \mathbf{0}^{|\mathcal{U}|-1}, 1)$ in this case.
– For each complete $\mathsf{QEnc}(\mathsf{pk}, \boldsymbol{x}^0_{\mathsf{pk}}, \boldsymbol{x}^1_{\mathsf{pk}}, \mathcal{U}, \ell_M)$ query,
  - $\mathcal{B}$ obtains $[\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}]_1 \leftarrow \mathsf{QSeedGen}(\mathsf{pk},\mathcal{U},\ell_M)$;
  - $\mathcal{B}$ obtains $(\mathsf{share}_{\mathsf{pk},\mathcal{U},\ell_M||\ell_K})_{\mathsf{pk}\in\mathcal{U}\cap\mathcal{H}} \leftarrow \mathsf{QShare}(\mathcal{U}, \ell_M||\ell_K)$;

  and implicitly complete the message in $\mathbb{G}_1$ as

$$[\boldsymbol{m}]_1 = [\boldsymbol{x}^b_{\mathsf{pk}}, \boldsymbol{x}^0_{\mathsf{pk}}, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}, \mathsf{share}_{\mathsf{pk},\mathcal{U},\ell_M||\ell_K} + {\boldsymbol{x}^b_{\mathsf{pk}}}^\top \cdot \boldsymbol{y}^b_{\mathsf{pk}}]$$

The admissibility condition $(*)$ of UZS holds since

– all the corruption queries in FH-IP-DDFE are sent in one shot;
– $\mathsf{QShare}(\mathcal{U}, \ell_M||\ell_K)$ queries are made for the same ℓ_K on every $\mathcal{U}$ while there are no $\mathsf{QTokGen}(\mathsf{pk},\mathcal{U},\ell_K)$ queries required.

When UZS.$b = 0$, one has $\mathsf{share}_{\mathsf{pk},\mathcal{U},\ell_M||\ell_K} = [\boldsymbol{a}^\top_{\mathsf{pk},\mathcal{U},\ell_M} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K}]_1$ which corresponds to $\mathbf{G}^*_{\ell_K.1}$; and when UZS.$b = 1$, one has $(\mathsf{share}_{\mathsf{pk},\mathcal{U},\ell_M||\ell_K})_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}} \xleftarrow{\$} \mathcal{R}_{\mathcal{H}\cap\mathcal{U},\ell_M||\ell_K}$ (as in Definition 16), which corresponds to $\mathbf{G}^*_{\ell_K.2}$. Therefore, one has

$$\left|\mathsf{Adv}_{\mathbf{G}^*_{\ell_K.1}} - \mathsf{Adv}_{\mathbf{G}^*_{\ell_K.2}}\right| \leq \mathsf{Adv}^{\text{otu-sta}}_{\mathsf{UZS}}.$$

□

Lemma 9 (FH-IP-DDFE: Transition from $\mathbf{G}^*_{\ell_K.2}$ to $\mathbf{G}^*_{\ell_K.3}$). *The two games $\mathbf{G}^*_{\ell_K.2}$ to $\mathbf{G}^*_{\ell_K.3}$ are identical.*

Proof. Given any set of complete encryption queries on $\{(\mathsf{pk}, \boldsymbol{x}^{\tau,0}_{\mathsf{pk}}, \boldsymbol{x}^{\tau,1}_{\mathsf{pk}}, \mathcal{U}, \ell_M)\}_{\tau\in[q_{\mathsf{pk},\mathcal{U},\ell_M}]}$ and any complete decryption-key query on $(\mathsf{pk}, \boldsymbol{y}^0_{\mathsf{pk}}, \boldsymbol{y}^1_{\mathsf{pk}}, \mathcal{U}, \ell_K)$, by the Remark 2, one has the following facts:

1. $\Delta^b_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K} = {\boldsymbol{x}^{\tau,0}}^\top_{\mathsf{pk}} \cdot \boldsymbol{y}^0_{\mathsf{pk}} - {\boldsymbol{x}^{\tau,b}}^\top_{\mathsf{pk}} \cdot \boldsymbol{y}^b_{\mathsf{pk}}$ $\forall \tau \in [q_{\mathsf{pk},\mathcal{U},\ell_M}]$;
2. $\sum_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}} \Delta^b_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K} = 0$.

For any random shares $(R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K})_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}}$ of the relation

$$\sum_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}} R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K} = -\sum_{\mathsf{pk}\in\mathcal{C}\cap\mathcal{U}} \boldsymbol{a}^\top_{\mathsf{pk},\mathcal{U},\ell_M} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_K},$$

one has $\left(R'_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K}\right)_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}} := \left(R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K} + \Delta^b_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K}\right)_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}}$ are shares of the same distribution by the fact 2. Moreover, one has the following by the fact 1,

$$\begin{aligned}
(\boldsymbol{x}^{\tau,b}_{\mathsf{pk}}, \boldsymbol{x}^{\tau,0}_{\mathsf{pk}}, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}, R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K} + {\boldsymbol{x}^{\tau,b}}^\top_{\mathsf{pk}} \cdot \boldsymbol{y}^b_{\mathsf{pk}}) &\overset{d}{=} (\boldsymbol{x}^{\tau,b}_{\mathsf{pk}}, \boldsymbol{x}^{\tau,0}_{\mathsf{pk}}, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}, R'_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K} + {\boldsymbol{x}^{\tau,b}}^\top_{\mathsf{pk}} \cdot \boldsymbol{y}^b_{\mathsf{pk}}) \\
&\overset{d}{=} (\boldsymbol{x}^{\tau,b}_{\mathsf{pk}}, \boldsymbol{x}^{\tau,0}_{\mathsf{pk}}, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}, R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K} + \Delta^b_{\ell_M,\mathsf{pk}} + {\boldsymbol{x}^{\tau,b}}^\top_{\mathsf{pk}} \cdot \boldsymbol{y}^b_{\mathsf{pk}}) \\
&\overset{d}{=} (\boldsymbol{x}^{\tau,b}_{\mathsf{pk}}, \boldsymbol{x}^{\tau,0}_{\mathsf{pk}}, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}, R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_K} + {\boldsymbol{x}^{\tau,0}}^\top_{\mathsf{pk}} \cdot \boldsymbol{y}^0_{\mathsf{pk}}).
\end{aligned}$$

Therefore, two games $\mathbf{G}^*_{\ell_K.2}$ and $\mathbf{G}^*_{\ell_K.3}$ identical. □

5 Attribute-Weighted-Sum DDFE

In this section, we construct a DDFE scheme for attribute-weighted sums from a UZS scheme, a single-input FE scheme for attribute-weighted sums with function-hiding inner products, and an all-or-nothing encapsulation AoNE scheme. The AWS-DDFE scheme is proved to be sel-sym-IND secure in the standard model.

5.1 Construction

Let d be an inner-product dimension, let $\mathcal{L}_M$ and $\mathcal{L}_K$ be a message-label space and a key-label space respectively. The AWS-DDFE scheme is described in Figure 12 with the following primitives:

- $\mathsf{AWIPE} = (\mathsf{aiSetup}, \mathsf{aiKeyGen}, \mathsf{aiEnc}, \mathsf{aiDKGen}, \mathsf{aiDec})$ be a FE for attribute-weighted sums with function-hiding inner products;
- $\mathsf{UZS} = (\mathsf{SetUp}, \mathsf{KeyGen}, \mathsf{SeedGen}, \mathsf{TokGen}, \mathsf{SeedUpt}, \mathsf{ShareEval})$ be a bilinear-updatable pseudorandom zero-sharing scheme over $\mathbb{G}_2$ for a seeding-label space $\mathcal{L}_K$ and an updating-label space $\mathcal{L}_M$.
- $\mathsf{AoNE} = (\mathsf{aSetup}, \mathsf{aKeyGen}, \mathsf{aEnc}, \mathsf{aDKGen}, \mathsf{aDec})$ be an all-or-nothing encapsulation scheme.

Correctness. Given $\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow \mathsf{SetUp}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}_{\mathsf{pk}}) \leftarrow \mathsf{KeyGen}()$ $\forall \mathsf{pk} \in \mathcal{PK}$, $\ell_M \in \mathcal{L}_M$, $\ell_{\boldsymbol{f}} \in \mathcal{L}_K$ where $\boldsymbol{f} = (f_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}$, $\mathcal{U}_M, \mathcal{U}_K \in \mathcal{S}(\mathcal{PK})$ such that $\mathcal{U}_M = \mathcal{U}_K = \mathcal{U}$, from the above scheme, one can parse $\mathsf{dk}_{\mathsf{pk}} = (\mathsf{act}_{\mathsf{pk}}, \mathcal{U}, \ell_{\boldsymbol{f}})$ and $\mathsf{ct}_{\mathsf{pk}} = (\mathsf{act}'_{\mathsf{pk}}, \mathcal{U}, \ell_M)$ for $\mathsf{pk} \in \mathcal{U}$. By the correctness of the AoNE scheme, one can always recover

$$\begin{aligned}
\mathsf{aict}_{\mathsf{pk}} &= \mathsf{aiEnc}(\mathsf{aisk}_{\mathsf{pk}}, (\boldsymbol{x}_{\mathsf{pk},j}, \boldsymbol{z}_{\mathsf{pk},j})_{j\in[N_{\mathsf{pk}}]}, [\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_M}, 0]_1); \\
\mathsf{aidk}_{\mathsf{pk}} &= \mathsf{aiKeyGen}(\mathsf{aisk}_{\mathsf{pk}}, f_{\mathsf{pk}}, [\boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}}, 0]_2).
\end{aligned}$$

The correctness is then implied by the correctness of the AWIPE scheme and the UZS scheme over $\mathbb{G}_2$:

$$\begin{aligned}
\sum_{\mathsf{pk}\in\mathcal{U}} \mathsf{aiDec}(\mathsf{aict}_{\mathsf{pk}}, \mathsf{aidk}_{\mathsf{pk}}) &= \sum_{\mathsf{pk}\in\mathcal{U}} [\sum_{j\in[N_{\mathsf{pk}}]} f_{\mathsf{pk}}(\boldsymbol{x}_{\mathsf{pk},j})^\top \cdot \boldsymbol{z}_{\mathsf{pk},j} + \boldsymbol{a}^\top_{\mathsf{pk},\mathcal{U},\ell_M} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_f}]_T \\
&= [\sum_{\mathsf{pk}\in\mathcal{U}} \sum_{j\in[N_{\mathsf{pk}}]} f_{\mathsf{pk}}(\boldsymbol{x}_{\mathsf{pk},j})^\top \cdot \boldsymbol{z}_{\mathsf{pk},j}]_T \\
&\quad + e\left([1]_1, \sum_{\mathsf{pk}\in\mathcal{U}} \mathsf{ShareEval}(\mathsf{SeedUpt}(\mathsf{seed}_{\mathsf{pk},\mathcal{U},\ell_f}, \mathsf{token}_{\mathsf{pk},\mathcal{U},\ell_M}))\right) \\
&= [\sum_{\mathsf{pk}\in\mathcal{U}} \sum_{j\in[N_{\mathsf{pk}}]} f_{\mathsf{pk}}(\boldsymbol{x}_{\mathsf{pk},j})^\top \cdot \boldsymbol{z}_{\mathsf{pk},j}]_T + e\left([1]_1, \sum_{\mathsf{pk}\in\mathcal{U}} \mathsf{share}_{\mathsf{pk},\mathcal{U},\ell_f||\ell_M}\right) \\
&= [\sum_{\mathsf{pk}\in\mathcal{U}} \sum_{j\in[N_{\mathsf{pk}}]} f_{\mathsf{pk}}(\boldsymbol{x}_{\mathsf{pk},j})^\top \cdot \boldsymbol{z}_{\mathsf{pk},j}]_T + e\left([1]_2, [0]_1\right) \\
&= [\sum_{\mathsf{pk}\in\mathcal{U}} \sum_{j\in[N_{\mathsf{pk}}]} f_{\mathsf{pk}}(\boldsymbol{x}_{\mathsf{pk},j})^\top \cdot \boldsymbol{z}_{\mathsf{pk},j}]_T.
\end{aligned}$$

□

Remark 7 (Size of Ciphertext/Decryption Key). In the above AWS-DDFE construction, if one uses the sel-sym-IND-secure AoNE that is constructed from a rate-1 identity-based encryption and employed in the hybrid-encryption mode with a symmetric encryption as described in [CDSG+20], then the complexity for the size each DDFE ciphertext/decryption key will be $O_\lambda(N + |\mathcal{U}|)$. Notably, N is the number of AWS inputs, which is polynomially unbounded.

5.2 Security Analysis

Theorem 3 (Indistinguishability for AWS-DDFE). *If* AWIPE *is a single-input* sel-sym-fh-IND*-secure FE for attribute-weighted sums with function-hiding inner products,* AoNE *is a* sel-sym-nfh-IND*-secure all-or-nothing encapsulation, and* UZS *is an* otu-sta-IND*-secure updatable pseudorandom zero sharing, then the* AWS-DDFE *scheme constructed in Figure 12 is* sel-sym-nfh-IND *secure (as defined in Definition 6) in the standard model.*

Proof. In the selective game, we can fix $\mathcal{PK}$ to be the set of parties generated by QNewHon() queries, $\mathcal{C}$ to be the set of corrupted parties in $\mathcal{PK}$ and $\mathcal{H} = \mathcal{PK} \setminus \mathcal{C}$ to be the set of honest parties. Let q_{xx} be the number of xx-oracle queries where $\mathrm{xx} \in \{\mathsf{QNewHon}, \mathsf{QEnc}, \mathsf{QDKGen}, \mathsf{QCor}\}$. For brevity, we use the notations $\hat{x} := (\boldsymbol{x}_j, \boldsymbol{z}_j)_{j\in[N]}$ and $\hat{f}$ for each ABP function f such that $\hat{f}(\hat{x}) := \sum_{j\in[N]} f(\boldsymbol{x}_j)^\top \boldsymbol{z}_j$. Given $\lambda \in \mathbb{N}$, we denote by $\mathsf{Adv}_{\mathbf{G}_i}$ the advantage of an PPT adversary $\mathcal{A}$ in each game $\mathbf{G}_i$, and $\mathsf{Adv}_{\mathrm{xx}}$ be the best advantage of any PPT adversary against the primitive xx that is setup with λ.

Since the UZS security applies only when there are more than one honest client, we consider two cases: only one honest client and more than one honest client.

The case of one honest client $\mathcal{H} = \{\mathsf{pk}^\star\}$. From the facts in Remark 4, any $\mathcal{U} \in \mathcal{S}(\mathcal{PK})$, any $\mathsf{QEnc}(\mathsf{pk}^\star, \hat{x}^0_{\mathsf{pk}^\star}, \hat{x}^1_{\mathsf{pk}^\star}, \mathcal{U}, \ell_M)$ query and any $\mathsf{QDKGen}(\mathsf{pk}^\star, (\hat{f}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}, \mathcal{U})$, then it must hold that

$$\hat{f}_{\mathsf{pk}^\star}(\hat{x}^0_{\mathsf{pk}^\star}) - \hat{f}_{\mathsf{pk}^\star}(\hat{x}^1_{\mathsf{pk}^\star}) = 0$$

This is also the admissibility condition of AWIPE. Following a strategy similar to the case of one honest client in Theorem 2, we construct a sequence of hybrid games:

$\mathbf{G}_0^{\mathsf{pk}^\star}$: This is the real game with one honest client $\mathsf{pk}^\star$.

$\mathbf{G}_1^{\mathsf{pk}^\star}$: The change is that the pseudorandom function $\mathsf{PRF}_{k_{\mathsf{pk}^\star}}$ is replaced by a random function RF. The indistinguishability is implied by the security of the PRF.

$\mathbf{G}_{1.\mathcal{U}}^{\mathsf{pk}^\star}$: For each $\mathcal{U}$, the change is that instead of depending on the bit b, for every $\mathsf{QEnc}(\mathsf{pk}, \hat{x}^0, \hat{x}^1, \mathcal{U}, \ell_M)$ query, the challenger chooses $\hat{x}^0$ to generate the answer.

Construction:

- $\mathsf{SetUp}(1^\lambda)$: Generates $\mathcal{PG} \leftarrow \mathsf{PGGen}(1^\lambda)$ and sets up parameters:

$$\mathsf{aipp} \leftarrow \mathsf{aiSetup}(1^\lambda) \quad \mathsf{upp} \leftarrow \mathsf{SetUp}(1^\lambda) \quad \mathsf{app} \leftarrow \mathsf{aSetup}(1^\lambda).$$

  It returns

$$\mathsf{pp} = (\mathcal{PG}, \mathsf{aipp}, \mathsf{upp}, \mathsf{app}).$$

  The parameters pp are implicit to other algorithms.
- $\mathsf{KeyGen}()$: Each client samples
  - a PRF key $k_{\mathsf{pk}} \xleftarrow{\$} \mathcal{K}_{\mathsf{PRF}}$;
  - UZS keys $(\mathsf{upk}, \mathsf{usk}_{\mathsf{pk}}) \leftarrow \mathsf{KeyGen}()$;
  - AoNE keys $(\mathsf{apk}, \mathsf{ask}_{\mathsf{pk}}) \leftarrow \mathsf{aKeyGen}()$.

  It returns $\mathsf{pk} = (\mathsf{upk}, \mathsf{apk})$ and $\mathsf{sk}_{\mathsf{pk}} = (k_{\mathsf{pk}}, \mathsf{usk}_{\mathsf{pk}}, \mathsf{ask}_{\mathsf{pk}})$.
- $\mathsf{Enc}(\mathsf{sk}_{\mathsf{pk}}, m)$: Parses $m = ((\boldsymbol{x}_j, \boldsymbol{z}_j)_{j\in[N]}, \mathcal{U}_M, \ell_M)$[a] and computes
  1. a UZS token: $\boldsymbol{b}_{\mathsf{pk},\mathcal{U}_M,\ell_M} \leftarrow \mathsf{TokGen}(\mathsf{usk}_{\mathsf{pk}}, \mathcal{U}_M, \ell_M)$;
  2. a random coin for AWIPE key generation: $\mathsf{coin}_{\mathsf{pk}} \leftarrow \mathsf{PRF}_{k_{\mathsf{pk}}}(\mathcal{U}_M)$;
  3. a AWIPE secret key: $\mathsf{aisk}_{\mathsf{pk}} = \mathsf{aiKeyGen}(1_{\mathsf{ip}}^{|\mathcal{U}_M|}; \mathsf{coin}_{\mathsf{pk}})$;
  4. an AWIPE encryption:

$$\mathsf{aict}_{\mathsf{pk}} \leftarrow \mathsf{aiEnc}(\mathsf{aisk}_{\mathsf{pk}}, (\boldsymbol{x}_j, \boldsymbol{z}_j)_{j\in[N]}, [\boldsymbol{b}_{\mathsf{pk},\mathcal{U}_M,\ell_M}, 0]_1);$$

  5. an AoNE layer on $\mathsf{ict}_{\mathsf{pk}}$:

$$\mathsf{act}_{\mathsf{pk}} \leftarrow \mathsf{aEnc}(\mathsf{ask}_{\mathsf{pk}}, (\mathsf{aict}_{\mathsf{pk}}, \mathcal{U}_M, \ell_M, \text{"}ct\text{"})).$$

  It returns the ciphertext

$$\mathsf{ct}_{\mathsf{pk}} = (\mathsf{act}_{\mathsf{pk}}, \mathcal{U}_M, \ell_M).$$

  If $\mathsf{pk} \notin \mathcal{U}_M$, it returns $\perp$.
- $\mathsf{DKGen}(\mathsf{sk}_{\mathsf{pk}}, k)$: Parses $k = (\boldsymbol{f} := (f_{\mathsf{pk}}, \mathsf{pk})_{\mathsf{pk}\in\mathcal{U}_K}, \mathcal{U}_K)$ and computes
  1. a UZS seed: $[\boldsymbol{a}_{\mathsf{pk},\mathcal{U}_K\ell_{\boldsymbol{f}}}]_2 \leftarrow \mathsf{SeedGen}(\mathsf{usk}_{\mathsf{pk}}, \mathcal{U}_K, \ell_{\boldsymbol{f}})$;[b]
  2. a random coin for AWIPE key generation: $\mathsf{coin}_{\mathsf{pk}} \leftarrow \mathsf{PRF}_{k_{\mathsf{pk}}}(\mathcal{U}_K)$;
  3. a AWIPE secret key: $\mathsf{aisk}_{\mathsf{pk}} = \mathsf{aiKeyGen}(1_{\mathsf{ip}}^{|\mathcal{U}_K|}; \mathsf{coin}_{\mathsf{pk}})$;
  4. an AWIPE decryption key:

$$\mathsf{aidk}_{\mathsf{pk}} \leftarrow \mathsf{aiDKGen}(\mathsf{aisk}_{\mathsf{pk}}, f_{\mathsf{pk}}, [\boldsymbol{a}_{\mathsf{pk},\ell_{\boldsymbol{f}}}, 0]_2);$$

  5. an AoNE layer on $\mathsf{aidk}_{\mathsf{pk}}$:

$$\mathsf{act}_{\mathsf{pk}} \leftarrow \mathsf{aEnc}(\mathsf{ask}_{\mathsf{pk}}, (\mathsf{aidk}_{\mathsf{pk}}, \mathcal{U}_K, \ell_{\boldsymbol{f}}, \text{"}dk\text{"})).$$

  It returns the decryption key

$$\mathsf{dk}_{\mathsf{pk}} = (\mathsf{act}_{\mathsf{pk}}, \mathcal{U}_K, \ell_{\boldsymbol{f}}).$$

  If $\mathsf{pk} \in \mathcal{U}_K$, it returns $\perp$.
- $\mathsf{Dec}((\mathsf{dk}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}_K}, (\mathsf{ct}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}_M}, (\mathcal{U}_M, \ell_M), (\mathcal{U}_K, \ell_{\boldsymbol{f}}))$: If $\mathcal{U}_M = \mathcal{U}_K = \mathcal{U}$ is not true, it returns $\perp$. Otherwise,
  1. it parses $\mathsf{dk}_{\mathsf{pk}} = (\mathsf{act}_{\mathsf{pk}}, \mathcal{U}, \ell_{\boldsymbol{f}})$ and recovers the AWIPE decryption keys

$$(\mathsf{aidk}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}} = \mathsf{aDec}((\mathsf{act}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}, \mathcal{U}, \ell_{\boldsymbol{f}});$$

  2. it parses $\mathsf{ct}_{\mathsf{pk}} = (\mathsf{act}'_{\mathsf{pk}}, \mathcal{U}, \ell_M)$ and recovers the AWIPE ciphertexts

$$(\mathsf{aict}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}} = \mathsf{aDec}((\mathsf{act}'_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}, \mathcal{U}, \ell_M);$$

  It returns $[\alpha]_T = \sum_{\mathsf{pk}\in\mathcal{U}} \mathsf{aiDec}(\mathsf{aict}_{\mathsf{pk}}, \mathsf{aidk}_{\mathsf{pk}})$.

[a] Each client can choose an arbitrary polynomial number N of AWS inputs.
[b] $\ell_{\boldsymbol{f}} \in \mathcal{L}_K$ contains a description of $\boldsymbol{f}$.

Fig. 12. DDFE for Attribute-Weighted Sums

The advantage of $\mathcal{A}$ is then upper bounded by $q_{\mathsf{QEnc}} \cdot \mathsf{Adv}_{\mathsf{AWIPE}} + \mathsf{Adv}_{\mathsf{PRF}}$.

The case of more than one honest client. Let $\mathcal{Q}_M$ and $\mathcal{Q}_K$ be the set of encryption queries and decryption key queries sent in one shot by $\mathcal{A}$ respectively. We proceed via a hybrid argument: we describe the global changes in the IND game by using the games $\mathbf{G}_0$, $\mathbf{G}_1$, $\mathbf{G}_2$ and $\mathbf{G}_3$; the transition between $\mathbf{G}_2$ and $\mathbf{G}_3$ requires intermediate games $(\mathbf{G}_{2.(\mathcal{U},\ell_M).i})_{i\in[5]}$ (see Figure 13) for each pair $(\mathcal{U},\ell_M) \in \mathcal{Q}_M$. Notably, the game $\mathbf{G}_0$ corresponds to sel-sym-nfh-IND security game as defined in Definition 6, and the game $\mathbf{G}_3$ corresponds to the case where adversary's advantage is 0 since there is no challenge bit b.

Game $\mathbf{G}_1$: The change is that the challenger uses a random function $\mathsf{RF}_{\mathsf{pk}}$ instead of $\mathsf{PRF}_{k_{\mathsf{pk}}}$ for $\mathsf{pk} \in \mathcal{H}$. The indistinguishability is implied by the security of the pseudorandom functions.

Game $\mathbf{G}_2$: When $\mathsf{pk} \in \mathcal{H}$, a decryption key query $(\mathsf{pk}, (\hat{f}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}_K}, \mathcal{U}_K) \in \mathcal{Q}_K$ is said to be incomplete if there exists $\mathsf{pk}' \in \mathcal{H} \cap \mathcal{U}_K$ and the key query $(\mathsf{pk}', (\hat{f}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}_K}, \mathcal{U}_K) \notin \mathcal{Q}_K$. For that query, $\mathsf{act}_{\mathsf{pk}}$ is changed to the encapsulation of $(0, \mathcal{U}_K, \ell_{\boldsymbol{f}}, \text{"}dk\text{"})$. Similarly, when $\mathsf{pk} \in \mathcal{H}$, an encryption query $(\mathsf{pk}, \hat{x}^0_{\mathsf{pk}}, \hat{x}^1_{\mathsf{pk}}, \mathcal{U}_M, \ell_M) \in \mathcal{Q}_M$ is said to be incomplete if there exists $\mathsf{pk}' \in \mathcal{H} \cap \mathcal{U}_M$ and the encryption query $(\mathsf{pk}', \hat{x}'^0_{\mathsf{pk}}, \hat{x}'^1_{\mathsf{pk}}, \mathcal{U}_M, \ell_M) \notin \mathcal{Q}_M$. For that query, $\mathsf{act}_{\mathsf{pk}}$ is changed to the encapsulation of $(0, \mathcal{U}_M, \ell_M, \text{"}ct\text{"})$. The indistinguishability is implied by the security of the AoNE scheme.

Game $\mathbf{G}_3$: In this game, for every complete encryption query on $(\mathsf{pk}, \hat{x}^0_{\mathsf{pk}}, \hat{x}^1_{\mathsf{pk}}, \mathcal{U}_M, \ell_M)$, the challenger sets the AWIPE message as $(\hat{x}^0_{\mathsf{pk}}, [\boldsymbol{b}_{\mathsf{pk},\mathcal{U}_M,\ell_M}, 0]_1)$. There is thus no dependence on the challenge bit b in this game, so $\mathsf{Adv}_{\mathbf{G}_3} = 0$.

To avoid duplicate arguments, we omit lemmas for the transitions from $\mathbf{G}_0$ to $\mathbf{G}_2$. The transitions in this stage are well-established and can be referenced in the proof for FH-IP-DDFE (see in Theorem 2 and Figure 11). Instead, we focus on the transition from $\mathbf{G}_2$ to $\mathbf{G}_3$, which is done by using the following intermediate games for each $(\mathcal{U},\ell_M) \in \mathcal{Q}_M$:

Game $\mathbf{G}_{2.(\mathcal{U},\ell_M).1}$: The change is that for every complete decryption key query on $(\mathsf{pk}, (\hat{f}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}, \mathcal{U})$, the challenger sets the AWIPE key as $(\hat{f}_{\mathsf{pk}}, [\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}}, \boldsymbol{a}^\top_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}} \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_M}]_2)$, and for every complete encryption query on $(\mathsf{pk}, \hat{x}^0_{\mathsf{pk}}, \hat{x}^1_{\mathsf{pk}}, \mathcal{U}, \ell_M)$, the challenger sets the AWIPE message as $(\hat{x}^b_{\mathsf{pk}}, [\mathbf{0}^{|\mathcal{U}|-1}, 1]_1)$. The indistinguishability is implied by the security of the AWIPE scheme, given in Lemma 10.

Game $\mathbf{G}_{2.(\mathcal{U},\ell_M).2}$: The change is that for every complete decryption key query on $(\mathsf{pk}, (\hat{f}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}, \mathcal{U})$, the challenger sets the AWIPE key as $(\hat{f}_{\mathsf{pk}}, [\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}}, R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_{\boldsymbol{f}}}]_2)$ where $(R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_{\boldsymbol{f}}})_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}}$ are sampled uniformly such that

$$\sum_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}} R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_{\boldsymbol{f}}} = \sum_{\mathsf{pk}\in\mathcal{C}\cap\mathcal{U}} \boldsymbol{a}^\top_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}} \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_M}.$$

The indistinguishability is implied by the security of the UZS scheme, given in Lemma 11.

Game $\mathbf{G}_{2.(\mathcal{U},\ell_M).3}$: The change is that given a set of $(\mathcal{U},\ell_M)$-involved complete encryption queries on $\{(\mathsf{pk}, \hat{x}^{\tau,0}_{\mathsf{pk}}, \hat{x}^{\tau,1}_{\mathsf{pk}}, \mathcal{U}, \ell_M)\}_{\tau\in[q_{\mathsf{pk},\mathcal{U},\ell_M}]}$, to answer any complete decryption-key query on $(\mathsf{pk}, (\hat{f}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}, \mathcal{U})$, the challenger sets the AWIPE key as $(\hat{f}_{\mathsf{pk}}, [\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}}, R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_{\boldsymbol{f}}} + \Delta^b_{\mathsf{pk},\mathcal{U},\ell_M,\ell_{\boldsymbol{f}}}]_2)$ where

$$\begin{aligned} \Delta^b_{\mathsf{pk},\mathcal{U},\ell_M,\ell_{\boldsymbol{f}}} &= \hat{f}_{\mathsf{pk}}(\hat{x}^{0,\tau}_{\mathsf{pk}}) - \hat{f}_{\mathsf{pk}}(\hat{x}^{b,\tau}_{\mathsf{pk}}) \ \forall \tau \in [q_{\mathsf{pk},\mathcal{U},\ell_M}], \\ \sum_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}} \Delta^b_{\mathsf{pk},\mathcal{U},\ell_M,\ell_{\boldsymbol{f}}} &= 0 \end{aligned}$$

as indicated in Remark 4. The indistinguishability is perfect, given in Lemma 12.

Game $\mathbf{G}_{2.(\mathcal{U},\ell_M).4}$: The change is that for every complete decryption-key query on $(\mathsf{pk}, (\hat{f}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}, \mathcal{U})$, the challenger sets the AWIPE key as $(\hat{f}_{\mathsf{pk}}, [\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}}, \boldsymbol{a}^\top_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}} \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_M}]_2)$. This change is symmetric to the change in $\mathbf{G}_{2.(\mathcal{U},\ell_M).2}$, and then the indistinguishability is implied by the security of the UZS scheme.

Game $\mathbf{G}_{2.(\mathcal{U},\ell_M)+1.0} := \mathbf{G}_{2.(\mathcal{U},\ell_M).5}$: For the subsequent pair $((\mathcal{U},\ell_M)+1)$ of $(\mathcal{U},\ell_M)$ in $\mathcal{Q}_M$, the change is that for every complete decryption key query on $(\mathsf{pk}, (\hat{f}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}, \mathcal{U})$, the challenger sets the AWIPE key as $(\hat{f}_{\mathsf{pk}}, [\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}}, 0]_2)$, and for every complete encryption query on $(\mathsf{pk}, \hat{x}^0_{\mathsf{pk}}, \hat{x}^1_{\mathsf{pk}}, \mathcal{U}, \ell_M)$, the challenger sets the AWIPE message as $(\hat{x}^b_{\mathsf{pk}}, [\boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_M}, 0]_1)$. The indistinguishability is implied by the security of the AWIPE scheme, given in Lemma 13.

By using a recursive transition through all $(\mathcal{U}, \ell_M) \in \mathcal{Q}_M$, one comes to the final game $\mathbf{G}_3$. One completes the theorem by having

$$\begin{aligned}\mathsf{Adv}^{\text{sel-sym-nfh}}_{\text{AWS-DDFE}} \leq\ & 2q_{\mathsf{QEnc}}(q_{\mathsf{QNewHon}} - q_{\mathsf{QCor}}) \cdot \mathsf{Adv}^{\text{sel-sym-fh}}_{\text{AWIPE}} + 2q_{\mathsf{QEnc}} \cdot \mathsf{Adv}^{\text{otu-sta}}_{\text{UZS}} \\ & + \mathsf{Adv}^{\text{sel-sym-nfh}}_{\text{AoNE}} + (q_{\mathsf{QNewHon}} - q_{\mathsf{QCor}}) \cdot \mathsf{Adv}_{\mathsf{PRF}}.\end{aligned}$$

□

Game	Adjustment	Assumption
$\mathbf{G}_{2.(\mathcal{U},\ell_M).0}$	aiEnc: $(\mathcal{U}', \ell'_M) < (\mathcal{U}, \ell_M)$: $(\hat{x}^0_{\mathsf{pk}}, \boldsymbol{b}_{\mathsf{pk},\mathcal{U}',\ell'_M}, 0)$ $(\mathcal{U}', \ell'_M) \geq (\mathcal{U}, \ell_M)$: $(\hat{x}^b_{\mathsf{pk}}, \boldsymbol{b}_{\mathsf{pk},\mathcal{U}',\ell'_M}, 0)$ aiDKGen: same as in $\mathbf{G}_2$	Hybrids on $(\mathcal{U}', \ell'_M) < (\mathcal{U}, \ell_M)$
$\mathbf{G}_{2.(\mathcal{U},\ell_M).1}$	aiEnc: $(\mathcal{U}', \ell'_M) = (\mathcal{U}, \ell_M) : (\hat{x}^b_{\mathsf{pk}}, \mathbf{0}^{\lvert U\rvert-1}, 1)$ aiKeyGen: $\mathcal{U}' = \mathcal{U}$: $(\hat{f}_{\mathsf{pk}}, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_f}, \boldsymbol{a}^\top_{\mathsf{pk},\mathcal{U},\ell_f} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_M})$	IND of AWIPE
$\mathbf{G}_{2.(\mathcal{U},\ell_M).2}$	aiEnc: same as in $\mathbf{G}_{2.(\mathcal{U},\ell_M).1}$ aiDKGen: $\mathcal{U}' = \mathcal{U}$: $(\hat{f}_{\mathsf{pk}}, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_f}, R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_f})$ where $\sum_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}} R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_f} = -\sum_{\mathsf{pk}\in\mathcal{C}\cap\mathcal{U}} \boldsymbol{a}^\top_{\mathsf{pk},\mathcal{U},\ell_f} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_M}$	IND of UZS
$\mathbf{G}_{2.(\mathcal{U},\ell_M).3}$	aiEnc: same as in $\mathbf{G}_{2.(\mathcal{U},\ell_M).1}$ aiDKGen: $\mathcal{U}' = \mathcal{U}$: $(\hat{f}_{\mathsf{pk}}, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_f}, R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_f} + \Delta^b_{\mathsf{pk},\mathcal{U},\ell_M,\ell_f})$ where $\sum_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}} R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_f} = -\sum_{\mathsf{pk}\in\mathcal{C}\cap\mathcal{U}} \boldsymbol{a}^\top_{\mathsf{pk},\mathcal{U},\ell_f} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_M}$ $\Delta^b_{\mathsf{pk},\mathcal{U},\ell_M,\ell_f} = \hat{f}_{\mathsf{pk}}(\hat{x}^0_{\mathsf{pk}}) - \hat{f}_{\mathsf{pk}}(\hat{x}^b_{\mathsf{pk}})$	Statistics
$\mathbf{G}_{2.(\mathcal{U},\ell_M).4}$	aiEnc: same as in $\mathbf{G}_{2.(\mathcal{U},\ell_M).1}$ aiDKGen: $\mathcal{U}' = \mathcal{U}$: $(\hat{f}_{\mathsf{pk}}, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_f}, \boldsymbol{a}^\top_{\mathsf{pk},\mathcal{U},\ell_f} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_M} + \Delta^b_{\mathsf{pk},\mathcal{U},\ell_M,\ell_f})$ where $\Delta^b_{\mathsf{pk},\mathcal{U},\ell_M,\ell_f} = \hat{f}_{\mathsf{pk}}(\hat{x}^0_{\mathsf{pk}}) - \hat{f}_{\mathsf{pk}}(\hat{x}^b_{\mathsf{pk}})$	IND of UZS
$\mathbf{G}_{2.(\mathcal{U},\ell_M)+1.0} :=$ $\mathbf{G}_{2.(\mathcal{U},\ell_M).5}$	aiEnc: $(\mathcal{U}', \ell'_M) = (\mathcal{U}, \ell_M) : (\hat{x}^0_{\mathsf{pk}}, \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_M}, 0)$ aiKeyGen: $\mathcal{U}' = \mathcal{U}$: $(\hat{f}_{\mathsf{pk}}, \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_f}, 0)$	IND of AWIPE

Fig. 13. Intermediate hybrids for the transition from $\mathbf{G}_2$ to $\mathbf{G}_3$ in Theorem 3. We denote by $(\mathcal{U}', \ell'_M) < (\mathcal{U}, \ell_M)$ when $(\mathcal{U}', \ell'_M)$ is a previous pair of $(\mathcal{U}, \ell_M)$ in the encryption-query set $\mathcal{Q}_M$.

Lemma 10 (AWS-DDFE: Transition from $\mathbf{G}_{2.(\mathcal{U},\ell_M).0}$ to $\mathbf{G}_{2.(\mathcal{U},\ell_M).1}$). *For any PPT adversary $\mathcal{A}$, the advantage in distinguishing two games is*

$$\left|\mathsf{Adv}_{\mathbf{G}_{2.(\mathcal{U},\ell_M).0}} - \mathsf{Adv}_{\mathbf{G}_{2.(\mathcal{U},\ell_M).1}}\right| \leq (q_{\mathsf{QNewHon}} - q_{\mathsf{QCor}}) \cdot \mathsf{Adv}^{\text{sel-sym-fh}}_{\text{AWIPE}}.$$

Proof. On a fixed user set $\mathcal{U}$, we proceed by using multiple hybrid games for each $\mathsf{pk} \in \mathcal{H}$. We build an adversary $\mathcal{B}$ against the sel-sym-fh security of AWIPE from an adversary $\mathcal{A}$ that distinguishes between two games in the transition. To simulate a AWS-DDFE challenger, $\mathcal{B}$ uses the AWIPE oracles to handle all AWIPE related operations for the reply of each $(\mathsf{pk}, \mathcal{U})$-involved query from $\mathcal{A}$.

- For each complete $\mathsf{QDKGen}(\mathsf{pk}, (\hat{f}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}, \mathcal{U})$ query, $\mathcal{B}$ prepares the AWIPE key as

$$k^0 = (\hat{f}_{\mathsf{pk}}, [\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}}, 0]_2),$$
$$k^1 = (\hat{f}_{\mathsf{pk}}, [\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}}, \boldsymbol{a}^{\top}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_M}]_2)$$

  where $\boldsymbol{f} = (\hat{f}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}$ and sends (pk, k^0, k^1) to the AWIPE decryption-key oracle. It uses the returned decryption key $\mathsf{aidk}_{\mathsf{pk}}$ to complete the reply to $\mathcal{A}$.
- For each complete $\mathsf{QEnc}(\mathsf{pk}, \hat{x}^0_{\mathsf{pk}}, \hat{x}^1_{\mathsf{pk}}, \mathcal{U}, \ell_M)$ query, $\mathcal{B}$ prepares the AWIPE message as

$$m^0 = (\hat{x}^b_{\mathsf{pk}}, [\boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_M}, 0]_1),$$
$$m^1 = (\hat{x}^b_{\mathsf{pk}}, [\boldsymbol{0}^{|\mathcal{U}|-1}, 1]_1);$$

  and sends (pk, m^0, m^1) to the AWIPE encryption oracle. It uses the returned ciphertext $\mathsf{aict}_{\mathsf{pk}}$ to complete the reply to $\mathcal{A}$.
- $\mathcal{B}$ outputs $\mathcal{A}$'s guess for the challenge bit AWIPE.b.

Let F_{AWIPE} be the functionality defined in Definition 10 for AWIPE. The admissibility condition $(*)$ of AWIPE in each transition holds since one always has

$$F_{\mathsf{AWIPE}}(k^0, m^0) = [\hat{f}_{\mathsf{pk}}(\hat{x}^b_{\mathsf{pk}}) + \boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}}{}^{\top} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_M}]_T = F_{\mathsf{AWIPE}}(k^1, m^1).$$

Therefore, in each transition of the multiple hybrid games for each $\mathsf{pk} \in \mathcal{H}$, when AWIPE.$b = 0$, $\mathcal{A}$ is playing the previous game; when AWIPE.$b = 1$, $\mathcal{A}$ is playing the subsequent game. Since the number of $\mathsf{pk} \in \mathcal{H}$ queried to either QEnc or QDKGen is bounded by $(q_{\mathsf{QNewHon}} - q_{\mathsf{QCor}})$, one has

$$\left|\mathsf{Adv}_{\mathbf{G}_{2.(\mathcal{U},\ell_M).0}} - \mathsf{Adv}_{\mathbf{G}_{2.(\mathcal{U},\ell_M).1}}\right| \leq (q_{\mathsf{QNewHon}} - q_{\mathsf{QCor}}) \cdot \mathsf{Adv}^{\text{sel-sym-fh}}_{\mathsf{AWIPE}}.$$

□

Lemma 11 (AWS-DDFE: Transition from $\mathbf{G}_{2.(\mathcal{U},\ell_M).1}$ to $\mathbf{G}_{2.(\mathcal{U},\ell_M).2}$). *For any PPT adversary $\mathcal{A}$, the advantage in distinguishing two games is*

$$\left|\mathsf{Adv}_{\mathbf{G}_{2.(\mathcal{U},\ell_M).1}} - \mathsf{Adv}_{\mathbf{G}_{2.(\mathcal{U},\ell_M).2}}\right| \leq \mathsf{Adv}^{\text{otu-sta}}_{\mathsf{UZS}}.$$

Proof. We build an adversary $\mathcal{B}$ against the otu-sta-IND security of UZS from an adversary $\mathcal{A}$ that distinguishes between two games in the transition. To simulate a AWS-DDFE challenger, $\mathcal{B}$ uses the UZS oracles to handle all UZS related operations.

- For each complete $\mathsf{QEnc}(\mathsf{pk}, \hat{x}^0_{\mathsf{pk}}, \hat{x}^1_{\mathsf{pk}}, \mathcal{U}', \ell'_M)$ query, $\mathcal{B}$ prepares the AWIPE message as
  - If $(\mathcal{U}', \ell'_M) \neq (\mathcal{U}, \ell_M)$: it obtains $\boldsymbol{b}_{\mathsf{pk},\mathcal{U}',\ell'_M} \leftarrow \mathsf{QTokGen}(\mathsf{pk}, \mathcal{U}', \ell'_M)$ to complete m.
  - If $(\mathcal{U}', \ell'_M) = (\mathcal{U}, \ell_M)$: it does not have to obtain $\boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_M}$ as $m = (\hat{x}^b, [\boldsymbol{0}^{|\mathcal{U}|-1}, 1]_1)$ in this case.
- For each complete $\mathsf{QDKGen}(\mathsf{pk}, (\hat{f}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}, \mathcal{U})$ query, $\mathcal{B}$ prepares the AWIPE key as
  - $\mathcal{B}$ obtains $[\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}}]_2 \leftarrow \mathsf{QSeedGen}(\mathsf{pk}, \mathcal{U}, \ell_{\boldsymbol{f}})$;
  - $\mathcal{B}$ obtains $(\mathsf{share}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}||\ell_M})_{\mathsf{pk}\in\mathcal{U}\cap\mathcal{H}} \leftarrow \mathsf{QShare}(\mathcal{U}, \ell_{\boldsymbol{f}}||\ell_M)$;

  where $\boldsymbol{f} = (f_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}$ and implicitly completes the key with the inner-product input in $\mathbb{G}_2$ as

$$k = (\hat{f}_{\mathsf{pk}}, [\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}}, \mathsf{share}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}||\ell_M}]_2).$$

The admissibility condition $(*)$ of UZS holds since

- all the corruption queries in AWS-DDFE are sent in one shot;
- $\mathsf{QShare}(\mathcal{U}, \ell_{\boldsymbol{f}}||\ell_M)$ queries are made for the same ℓ_M on every $\mathcal{U}$ while there are no $\mathsf{QTokGen}(\mathsf{pk}, \mathcal{U}, \ell_M)$ queries required.

When $\mathsf{UZS}.b = 0$, one has $\mathsf{share}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}||\ell_M} = [\boldsymbol{a}^{\top}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_M}]_2$ which corresponds to $\mathbf{G}_{2.(\mathcal{U},\ell_M).1}$; and when $\mathsf{UZS}.b = 1$, one has $(\mathsf{share}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}||\ell_M})_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}} \xleftarrow{\$} \mathcal{R}_{\mathcal{H}\cap\mathcal{U},\ell_{\boldsymbol{f}}||\ell_M}$ (as defined in Definition 16), which corresponds to $\mathbf{G}_{2.(\mathcal{U},\ell_M).2}$. Therefore, one has

$$\left|\mathsf{Adv}_{\mathbf{G}_{2.(\mathcal{U},\ell_M).1}} - \mathsf{Adv}_{\mathbf{G}_{2.(\mathcal{U},\ell_M).2}}\right| \leq \mathsf{Adv}^{\text{otu-sta}}_{\mathsf{UZS}}.$$

□

Lemma 12 (AWS-DDFE: Transition from $\mathbf{G}_{2.(\mathcal{U},\ell_M).2}$ to $\mathbf{G}_{2.(\mathcal{U},\ell_M).3}$). *The two games $\mathbf{G}_{2.(\mathcal{U},\ell_M).2}$ to $\mathbf{G}_{2.(\mathcal{U},\ell_M).3}$ are identical.*

Proof. From the fact that $\sum_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}} \Delta^b_{\mathsf{pk},\mathcal{U},\ell_M,\ell_{\boldsymbol{f}}} = 0$, for any random shares $\left(R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_{\boldsymbol{f}}}\right)_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}}$ of the relation

$$\sum_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}} R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_{\boldsymbol{f}}} = -\sum_{\mathsf{pk}\in\mathcal{C}\cap\mathcal{U}} \boldsymbol{a}^{\top}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_M},$$

one has $\left(R_{\mathsf{pk},\mathcal{U},\ell_M,\ell_{\boldsymbol{f}}} + \Delta^b_{\mathsf{pk},\mathcal{U},\ell_M,\ell_{\boldsymbol{f}}}\right)_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}}$ are also random shares of the same relation. □

Lemma 13 (AWS-DDFE: Transition from $\mathbf{G}_{2.(\mathcal{U},\ell_M).4}$ to $\mathbf{G}_{2.(\mathcal{U},\ell_M).5}$). *For any PPT adversary $\mathcal{A}$, the advantage in distinguishing two games is*

$$\left|\mathsf{Adv}_{\mathbf{G}_{2.(\mathcal{U},\ell_M).4}} - \mathsf{Adv}_{\mathbf{G}_{2.(\mathcal{U},\ell_M).5}}\right| \leq (q_{\mathsf{QNewHon}} - q_{\mathsf{QCor}}) \cdot \mathsf{Adv}^{\text{sel-sym-fh}}_{\mathsf{AWIPE}}.$$

Proof. For any $\mathsf{pk} \in \mathcal{H}$, given a set of complete encryption queries on $\{(\mathsf{pk}, \hat{x}^{\tau,0}_{\mathsf{pk}}, \hat{x}^{\tau,1}_{\mathsf{pk}}, \mathcal{U}, \ell_M)\}_{\tau\in[q_{\mathsf{pk},\mathcal{U},\ell_M}]}$ that share the same $(\mathcal{U}, \ell_M)$ and any complete decryption-key query on $(\mathsf{pk}, (\hat{f}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}, \mathcal{U})$, one has the following facts by the Remark 4:

1. $\Delta^b_{\mathsf{pk},\mathcal{U},\ell_M,\ell_{\boldsymbol{f}}} = \hat{f}_{\mathsf{pk}}(\hat{x}^{0,\tau}_{\mathsf{pk}}) - \hat{f}_{\mathsf{pk}}(\hat{x}^{b,\tau}_{\mathsf{pk}})\ \forall\tau \in [q_{\mathsf{pk},\mathcal{U},\ell_M}]$,
2. $\sum_{\mathsf{pk}\in\mathcal{H}\cap\mathcal{U}} \Delta^b_{\mathsf{pk},\mathcal{U},\ell_M,\ell_{\boldsymbol{f}}} = 0$.

We proceed by using multiple hybrid games for each $\mathsf{pk} \in \mathcal{H}$. We build an adversary $\mathcal{B}$ against the sel-sym-fh security of AWIPE from an adversary $\mathcal{A}$ that distinguishes between two games in the transition. To simulate a AWS-DDFE challenger, $\mathcal{B}$ uses the AWIPE oracles to handle all AWIPE related operations for the reply of each $(\mathsf{pk},\mathcal{U})$-involved query from $\mathcal{A}$.

- For each complete $\mathsf{QDKGen}(\mathsf{pk}, (\hat{f}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}, \mathcal{U})$ query, $\mathcal{B}$ prepares the AWIPE key as

$$k^0 = (\hat{f}_{\mathsf{pk}}, [\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}}, \boldsymbol{a}^{\top}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_M} + \Delta^b_{\mathsf{pk},\mathcal{U},\ell_M,\ell_{\boldsymbol{f}}}]_2)$$
$$k^1 = (\hat{f}_{\mathsf{pk}}, [\boldsymbol{a}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}}, \boldsymbol{a}^{\top}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_M}]_2)$$

  where $\boldsymbol{f} = (f_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}$ and sends (pk, k^0, k^1) to the AWIPE decryption key generation oracle. It uses the returned decryption key $\mathsf{aidk}_{\mathsf{pk}}$ to complete the reply to $\mathcal{A}$.
- For each complete $\mathsf{QEnc}(\mathsf{pk}, \hat{x}^0_{\mathsf{pk}}, \hat{x}^1_{\mathsf{pk}}, \mathcal{U}, \ell_M)$ query, $\mathcal{B}$ prepares the AWIPE message as

$$m^0 = (\hat{x}^b_{\mathsf{pk}}, [\mathbf{0}^{|\mathcal{U}|-1}, 1]_1);$$
$$m^1 = (\hat{x}^0_{\mathsf{pk}}, [\boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_M}, 0]_1);$$

  and sends $(\mathsf{pk}, \hat{m}^0, \hat{m}^1)$ to the AWIPE encryption oracle. It uses the returned ciphertext $\mathsf{aict}_{\mathsf{pk}}$ to complete the reply to $\mathcal{A}$.
- $\mathcal{B}$ outputs $\mathcal{A}$'s guess for the challenge bit $\mathsf{AWIPE}.b$.

Let F_{AWIPE} be the functionality defined in Definition 10 for AWIPE. The admissibility condition $(*)$ of AWIPE in each transition holds since one always has

$$\begin{aligned} F_{\mathsf{AWIPE}}(k^0, m^0) &= [(\hat{f}_{\mathsf{pk}}(\hat{x}^b_{\mathsf{pk}}) + \Delta^b_{\mathsf{pk},\mathcal{U},\ell_M,\ell_{\boldsymbol{f}}}) + \boldsymbol{a}^{\top}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_M}]_T \\ &= [\hat{f}_{\mathsf{pk}}(\hat{x}^0_{\mathsf{pk}}) + \boldsymbol{a}^{\top}_{\mathsf{pk},\mathcal{U},\ell_{\boldsymbol{f}}} \cdot \boldsymbol{b}_{\mathsf{pk},\mathcal{U},\ell_M}]_T \text{ (by the above fact 1)} \\ &= F_{\mathsf{AWIPE}}(k^1, m^1) \end{aligned}$$

The index τ is omitted in the above equalities as $\Delta^b_{\mathsf{pk},\mathcal{U},\ell_M,\ell_{\boldsymbol{f}}}$ applies to all pairs of $\tau \in [q_{\mathsf{pk},\mathcal{U},\ell_M}]$ and $\boldsymbol{f} = (\hat{f}_{\mathsf{pk}})_{\mathsf{pk}\in\mathcal{U}}$. Therefore, in each transition of the multiple hybrid games for each $\mathsf{pk} \in \mathcal{H}$, when $\mathsf{AWIPE}.b = 0$, $\mathcal{A}$ is playing the previous game; when $\mathsf{AWIPE}.b = 1$, $\mathcal{A}$ is playing the subsequent game. Since the number of pairs $\mathsf{pk} \in \mathcal{H}$ queried to either QEnc or QDKGen is bounded by $(q_{\mathsf{QNewHon}} - q_{\mathsf{QCor}})$, one has

$$\left|\mathsf{Adv}_{\mathbf{G}_{2.(\mathcal{U},\ell_M).4}} - \mathsf{Adv}_{\mathbf{G}_{2.(\mathcal{U},\ell_M).5}}\right| \leq (q_{\mathsf{QNewHon}} - q_{\mathsf{QCor}}) \cdot \mathsf{Adv}^{\text{sel-sym-fh}}_{\mathsf{AWIPE}}.$$

□

Acknowledgements. We would like to thank Ky Nguyen, Duong Hieu Phan and David Pointcheval for fruitful discussions that motivated this work. The project was supported by the PhD funding from "Institut Polytechnique de Paris".

References

ABDP15. M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval. Simple functional encryption schemes for inner products. In *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*, *Lecture Notes in Computer Science* 9020, pages 733–751, Gaithersburg, MD, USA, March 30 – April 1, 2015. Springer, Heidelberg, Germany.

ABG19. M. Abdalla, F. Benhamouda, and R. Gay. From single-input to multi-client inner-product functional encryption. In *Advances in Cryptology – ASIACRYPT 2019, Part III*, *Lecture Notes in Computer Science* 11923, pages 552–582, Kobe, Japan, December 8–12, 2019. Springer, Heidelberg, Germany.

ABKW19. M. Abdalla, F. Benhamouda, M. Kohlweiss, and H. Waldner. Decentralizing inner-product functional encryption. In *PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part II*, *Lecture Notes in Computer Science* 11443, pages 128–157, Beijing, China, April 14–17, 2019. Springer, Heidelberg, Germany.

ABM+20. M. Abdalla, F. Bourse, H. Marival, D. Pointcheval, A. Soleimanian, and H. Waldner. Multi-client inner-product functional encryption in the random-oracle model. In *SCN 20: 12th International Conference on Security in Communication Networks*, *Lecture Notes in Computer Science* 12238, pages 525–545, Amalfi, Italy, September 14–16, 2020. Springer, Heidelberg, Germany.

ACF+18. M. Abdalla, D. Catalano, D. Fiore, R. Gay, and B. Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In *Advances in Cryptology – CRYPTO 2018, Part I*, *Lecture Notes in Computer Science* 10991, pages 597–627, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

ACF+20. S. Agrawal, M. Clear, O. Frieder, S. Garg, A. O'Neill, and J. Thaler. Ad hoc multi-input functional encryption. In *ITCS 2020: 11th Innovations in Theoretical Computer Science Conference*, pages 40:1–40:41, Seattle, WA, USA, January 12–14, 2020. LIPIcs.

ACGU20. M. Abdalla, D. Catalano, R. Gay, and B. Ursu. Inner-product functional encryption with fine-grained access control. In *Advances in Cryptology – ASIACRYPT 2020, Part III*, *Lecture Notes in Computer Science* 12493, pages 467–497, Daejeon, South Korea, December 7–11, 2020. Springer, Heidelberg, Germany.

AGT21a. S. Agrawal, R. Goyal, and J. Tomida. Multi-input quadratic functional encryption from pairings. In *Advances in Cryptology – CRYPTO 2021, Part IV*, *Lecture Notes in Computer Science* 12828, pages 208–238, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany.

AGT21b. S. Agrawal, R. Goyal, and J. Tomida. Multi-party functional encryption. In *TCC 2021: 19th Theory of Cryptography Conference, Part II*, *Lecture Notes in Computer Science* 13043, pages 224–255, Raleigh, NC, USA, November 8–11, 2021. Springer, Heidelberg, Germany.

AGT22. S. Agrawal, R. Goyal, and J. Tomida. Multi-input quadratic functional encryption: Stronger security, broader functionality. In *TCC 2022: 20th Theory of Cryptography Conference, Part I*, *Lecture Notes in Computer Science* 13747, pages 711–740, Chicago, IL, USA, November 7–10, 2022. Springer, Heidelberg, Germany.

AGW20. M. Abdalla, J. Gong, and H. Wee. Functional encryption for attribute-weighted sums from k-Lin. In *Advances in Cryptology – CRYPTO 2020, Part I*, *Lecture Notes in Computer Science* 12170, pages 685–716, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany.

AJ15. P. Ananth and A. Jain. Indistinguishability obfuscation from compact functional encryption. In *Advances in Cryptology – CRYPTO 2015, Part I*, *Lecture Notes in Computer Science* 9215, pages 308–326, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

ALMT20. S. Agrawal, B. Libert, M. Maitra, and R. Titiu. Adaptive simulation security for inner product functional encryption. In *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part I*, *Lecture Notes in Computer Science* 12110, pages 34–64, Edinburgh, UK, May 4–7, 2020. Springer, Heidelberg, Germany.

ALS16. S. Agrawal, B. Libert, and D. Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In *Advances in Cryptology – CRYPTO 2016, Part III*, *Lecture Notes in Computer Science* 9816, pages 333–362, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

ATY23. S. Agrawal, J. Tomida, and A. Yadav. Attribute-based multi-input FE (and more) for attribute-weighted sums. In *Advances in Cryptology – CRYPTO 2023, Part IV*, *Lecture Notes in Computer Science* 14084, pages 464–497, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Heidelberg, Germany.

BCFG17. C. E. Z. Baltico, D. Catalano, D. Fiore, and R. Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In *Advances in Cryptology – CRYPTO 2017, Part I*, *Lecture Notes in Computer Science* 10401, pages 67–98, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

BF01. D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology – CRYPTO 2001*, *Lecture Notes in Computer Science* 2139, pages 213–229, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

BIK+17. K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 1175–1191, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press.

BJK15. A. Bishop, A. Jain, and L. Kowalczyk. Function-hiding inner product encryption. In *Advances in Cryptology – ASIACRYPT 2015, Part I*, *Lecture Notes in Computer Science* 9452, pages 470–491, Auckland, New Zealand, November 30 – December 3, 2015. Springer, Heidelberg, Germany.

BSW11. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *TCC 2011: 8th Theory of Cryptography Conference*, *Lecture Notes in Computer Science* 6597, pages 253–273, Providence, RI, USA, March 28–30, 2011. Springer, Heidelberg, Germany.

BV15. N. Bitansky and V. Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *56th Annual Symposium on Foundations of Computer Science*, pages 171–190, Berkeley, CA, USA, October 17–20, 2015. IEEE Computer Society Press.

BW07. D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC 2007: 4th Theory of Cryptography Conference*, *Lecture Notes in Computer Science* 4392, pages 535–554, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Heidelberg, Germany.

CDG+18. J. Chotard, E. Dufour Sans, R. Gay, D. H. Phan, and D. Pointcheval. Decentralized multi-client functional encryption for inner product. In *Advances in Cryptology – ASIACRYPT 2018, Part II*, *Lecture Notes in Computer Science* 11273, pages 703–732, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany.

CDSG+20. J. Chotard, E. Dufour-Sans, R. Gay, D. H. Phan, and D. Pointcheval. Dynamic decentralized functional encryption. In *Advances in Cryptology – CRYPTO 2020, Part I*, *Lecture Notes in Computer Science* 12170, pages 747–775, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany.

Cha07. M. Chase. Multi-authority attribute based encryption. In *TCC 2007: 4th Theory of Cryptography Conference*, *Lecture Notes in Computer Science* 4392, pages 515–534, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Heidelberg, Germany.

CLT18. G. Castagnos, F. Laguillaumie, and I. Tucker. Practical fully secure unrestricted inner product functional encryption modulo p. In *Advances in Cryptology – ASIACRYPT 2018, Part II*, *Lecture Notes in Computer Science* 11273, pages 733–764, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany.

DP19. E. Dufour Sans and D. Pointcheval. Unbounded inner-product functional encryption with succinct keys. In *ACNS 19: 17th International Conference on Applied Cryptography and Network Security*, *Lecture Notes in Computer Science* 11464, pages 426–441, Bogota, Colombia, June 5–7, 2019. Springer, Heidelberg, Germany.

GGG+14. 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 *Advances in Cryptology – EUROCRYPT 2014*, *Lecture Notes in Computer Science* 8441, pages 578–602, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

GKL+13. S. D. Gordon, J. Katz, F.-H. Liu, E. Shi, and H.-S. Zhou. Multi-input functional encryption. Cryptology ePrint Archive, Report 2013/774, 2013. `https://eprint.iacr.org/2013/774`.

GPSW06. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS 2006: 13th Conference on Computer and Communications*

Security, pages 89–98, Alexandria, Virginia, USA, October 30 – November 3, 2006. ACM Press. Available as Cryptology ePrint Archive Report 2006/309.

IW14. Y. Ishai and H. Wee. Partial garbling schemes and their applications. In *ICALP 2014: 41st International Colloquium on Automata, Languages and Programming, Part I*, *Lecture Notes in Computer Science* 8572, pages 650–662, Copenhagen, Denmark, July 8–11, 2014. Springer, Heidelberg, Germany.

KSW08. J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Advances in Cryptology – EUROCRYPT 2008*, *Lecture Notes in Computer Science* 4965, pages 146–162, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany.

LT19. B. Libert and R. Titiu. Multi-client functional encryption for linear functions in the standard model from LWE. In *Advances in Cryptology – ASIACRYPT 2019, Part III*, *Lecture Notes in Computer Science* 11923, pages 520–551, Kobe, Japan, December 8–12, 2019. Springer, Heidelberg, Germany.

LW11. A. B. Lewko and B. Waters. Decentralizing attribute-based encryption. In *Advances in Cryptology – EUROCRYPT 2011*, *Lecture Notes in Computer Science* 6632, pages 568–588, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany.

LWG+23. Y. Li, J. Wei, F. Guo, W. Susilo, and X. Chen. Robust decentralized multi-client functional encryption: Motivation, definition, and inner-product constructions. In *Advances in Cryptology – ASIACRYPT 2023, Part V*, *Lecture Notes in Computer Science* 14442, pages 134–165, Guangzhou, China, December 4–8, 2023. Springer, Heidelberg, Germany.

MJ18. Y. Michalevsky and M. Joye. Decentralized policy-hiding ABE with receiver privacy. In *ESORICS 2018: 23rd European Symposium on Research in Computer Security, Part II*, *Lecture Notes in Computer Science* 11099, pages 548–567, Barcelona, Spain, September 3–7, 2018. Springer, Heidelberg, Germany.

MKMS22. J. M. B. Mera, A. Karmakar, T. Marc, and A. Soleimanian. Efficient lattice-based inner-product functional encryption. In *PKC 2022: 25th International Conference on Theory and Practice of Public Key Cryptography, Part II*, *Lecture Notes in Computer Science* 13178, pages 163–193, Virtual Event, March 8–11, 2022. Springer, Heidelberg, Germany.

NPP22. K. Nguyen, D. H. Phan, and D. Pointcheval. Multi-client functional encryption with fine-grained access control. In *Advances in Cryptology – ASIACRYPT 2022, Part I*, *Lecture Notes in Computer Science* 13791, pages 95–125, Taipei, Taiwan, December 5–9, 2022. Springer, Heidelberg, Germany.

NPP23a. D. D. Nguyen, D. H. Phan, and D. Pointcheval. Verifiable decentralized multi-client functional encryption for inner product. In *Advances in Cryptology – ASIACRYPT 2023, Part V*, *Lecture Notes in Computer Science* 14442, pages 33–65, Guangzhou, China, December 4–8, 2023. Springer, Heidelberg, Germany.

NPP23b. K. Nguyen, D. H. Phan, and D. Pointcheval. Optimal security notion for decentralized multi-client functional encryption. In *ACNS 23: 21st International Conference on Applied Cryptography and Network Security, Part II*, *Lecture Notes in Computer Science* 13906, pages 336–365, Kyoto, Japan, June 19–22, 2023. Springer, Heidelberg, Germany.

Sha84. A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology – CRYPTO'84*, *Lecture Notes in Computer Science* 196, pages 47–53, Santa Barbara, CA, USA, August 19–23, 1984. Springer, Heidelberg, Germany.

SV23. E. Shi and N. Vanjani. Multi-client inner product encryption: Function-hiding instantiations without random oracles. In *PKC 2023: 26th International Conference on Theory and Practice of Public Key Cryptography, Part I*, *Lecture Notes in Computer Science* 13940, pages 622–651, Atlanta, GA, USA, May 7–10, 2023. Springer, Heidelberg, Germany.

SW05. A. Sahai and B. R. Waters. Fuzzy identity-based encryption. In *Advances in Cryptology – EUROCRYPT 2005*, *Lecture Notes in Computer Science* 3494, pages 457–473, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany.

TT18. J. Tomida and K. Takashima. Unbounded inner product functional encryption from bilinear maps. In *Advances in Cryptology – ASIACRYPT 2018, Part II*, *Lecture Notes in Computer Science* 11273, pages 609–639, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany.

ZLZ+24. Z. Zhu, J. Li, K. Zhang, J. Gong, and H. Qian. Registered functional encryptions from pairings. Cryptology ePrint Archive, Paper 2024/327, 2024. `https://eprint.iacr.org/2024/327`.