Designs, Codes and Cryptography
https://doi.org/10.1007/s10623-024-01432-8

Compact FE for unbounded attribute-weighted sums for logspace from SXDH

Pratish Datta[1] · Tapas Pal[2] · Katsuyuki Takashima[3]

Received: 10 August 2023 / Revised: 27 February 2024 / Accepted: 20 May 2024
© The Author(s) 2024

Abstract
This paper presents the *first* functional encryption (FE) scheme for the attribute-weighted sum functionality that supports the *uniform* model of computation. In such an FE scheme, encryption takes as input a pair of attributes (x, z) where x is public and z is private. A secret key corresponds to some weight function f, and decryption recovers the weighted sum $f(x)z$. In our scheme, both the public and private attributes can be of arbitrary polynomial lengths that are not fixed at system setup. The weight functions are modelled as Logspace Turing machines. Prior schemes could only support non-uniform Logspace. The proposed scheme is proven *adaptively simulation* secure under the well-studied symmetric external Diffie–Hellman assumption against an arbitrary polynomial number of secret key queries both before and after the challenge ciphertext. This is the best possible security notion that could be achieved for FE. On the technical side, our contributions lie in extending the techniques of Lin and Luo [EUROCRYPT 2020] devised for indistinguishability-based payload hiding attribute-based encryption for uniform Logspace access policies and the "three-slot reduction" technique for simulation-secure attribute-hiding FE for non-uniform Logspace devised by Datta and Pal [ASIACRYPT 2021] to the context of simulation-secure attribute-hiding FE for uniform Logspace.

Keywords Functional encryption · Attribute-weighted sums · Logspace · Turing machines

Communicated by C. Weinert.

This is the full version of an extended abstract that has appeared in ASIACRYPT 2022.

T. Pal: This work was done when the author was a postdoctoral researcher at NTT Social Informatics Laboratories, Japan.

✉ Tapas Pal
tapas.pal@kit.edu

Pratish Datta
pratish.datta@ntt-research.com

Katsuyuki Takashima
ktakashima@waseda.jp

1 NTT Research, Sunnyvale, CA 94085, USA

2 Karlsruhe Institute of Technology, KASTEL Security Research Labs, Karlsruhe 76131, Germany

3 Waseda University, Shinjuku-ku, Tokyo 169-8050, Japan

Mathematics Subject Classification 94A60

1 Introduction

Functional encryption (FE), formally introduced by Boneh et al. [24] and O'Neill [69], redefines the classical encryption procedure with the motivation to overcome the limitation of the "all-or-nothing" paradigm of decryption. In a traditional encryption system, there is a single secret key such that a user given a ciphertext can either recover the whole message or learns nothing about it, depending on the availability of the secret key. FE in contrast provides fine grained access control over encrypted data by generating artistic secret keys according to the desired functions of the encrypted data to be disclosed. More specifically, in a public-key FE scheme for a function class $\mathcal{F}$, there is a setup authority which produces a master secret key and publishes a master public key. Using the master secret key, the setup authority can derive secret keys or functional decryption keys SK_f associated with functions $f \in \mathcal{F}$. Anyone can encrypt messages msg belonging to a specified message space $\mathsf{msg} \in \mathbb{M}$ using the master public key to produce a ciphertext CT. The ciphertext CT along with a secret key SK_f recovers the function of the message $f(\mathsf{msg})$ at the time of decryption, while unable to extract any other information about msg. More specifically, the security of FE requires *collusion resistance* meaning that any polynomial number of secret keys together cannot gather more information about an encrypted message except the union of what each of the secret keys can learn individually.

By this time, we have a plethora of exciting works on FE. These works can be broadly classified in two categories. The first line of works attempted to build FE for general functionalities [12–17, 20, 23, 25–28, 34, 41–50, 52, 53, 55, 60, 74]. However, those constructions were either only secure against bounded collusion and/or extremely inefficient. With the motivation to overcome these limitations, a second line of work attempted to design efficient FE schemes supporting arbitrary collusion of users for practically relevant functionalities, e.g., linear/quadratic functions [1–11, 21, 29, 32, 33, 35, 36, 38, 40, 54, 56, 61, 63–66, 70–72, 76, 77]. In this work, we advance the state of the art along the latter research direction.

FE **for Attribute-Weighted Sum** Recently, Abdalla et al. [8] and Datta and Pal [38] studied FE schemes for a new class of functionalities termed as "attribute-weighted sums" (AWS). This is a generalization of the inner product functional encryption (IPFE) [3, 11]. In such a scheme, an attribute pair (x, z) is encrypted using the master public key of the scheme, where x is a public attribute (e.g., demographic data) and z is a private attribute containing sensitive information (e.g., salary, medical condition, loans, college admission outcomes). A recipient having a secret key corresponding to a weight function f can learn the attribute-weighted sum $f(x)z$. The attribute-weighted sum functionality appears naturally in several real life applications. For instance, as discussed by Abdalla et al. [8] if we consider the weight function f as a Boolean predicate, then the attribute-weighted sum functionality $f(x)$ would correspond to the average z over all users whose attribute x satisfies the predicate f. Important practical scenarios include average salaries of minority groups holding a particular job ($z =$ salary) and approval ratings of an election candidate amongst specific demographic groups in a particular state ($z =$ rating).

The works of [8, 38] considered a more general case of the notion where the domain and range of the weight functions are vectors, in particular, the attribute pair of public/private attribute vectors $(\boldsymbol{x}, \boldsymbol{z})$, which is encrypted to a ciphertext CT. A secret key SK_f generated for a weight function f allows a recipient to learn $f(\boldsymbol{x})^{\top}\boldsymbol{z}$ from CT without leaking any information about the private attribute $\boldsymbol{z}$.

The FE schemes of [8, 38] support an expressive function class of *arithmetic branching programs* (ABPs) which captures non-uniform Logspace computations. Both schemes were built in asymmetric bilinear groups of prime order and are proven secure in the simulation-based security model, which is known to be the desirable security model for FE [24, 69], under the (bilateral) k-Linear (k-Lin)/(bilateral) *Matrix Diffie–Hellman* (MDDH) assumption. The FE scheme of [8] achieves semi-adaptive security, where the adversary is restricted to making secret key queries only after making the ciphertext queries, whereas the FE scheme of [38] achieves adaptive security, where the adversary is allowed to make secret key queries both before and after the ciphertext queries.

However, as mentioned above, ABP is a non-uniform computational model. As such, in both the FE schemes [8, 38], the length of the public and private attribute vectors must be fixed at system setup. This is clearly a bottleneck in several applications of this primitive especially when the computation is done over attributes whose lengths vary widely among ciphertexts and are not fixed at system setup. For instance, suppose a government hires an external audit service to perform a survey on average salary of employees working under different job categories in various companies to resolve salary discrepancy. The companies create salary databases (X, Z) where $X = (x_i)_i$ contains public attributes $x_i = (\text{job title, department, company name})$ and $Z = (z_i)_i$ includes private attribute $z_i = \text{salary}$. To facilitate this auditing process without revealing individual salaries (private attribute) to the auditor, the companies encrypt their own database (X, Z) using an FE scheme for AWS. The government provides the auditor a functional secret key SK_f for a function f that takes input a public attribute X and outputs 1 for x_i's for which the "job title" matches with a particular job, say *manager*. The auditor decrypts ciphertexts of the various companies using SK_f and calculates the average salaries of employees working under that job category in those companies. Now, if the existing FE schemes for AWS [8, 38] supporting non-uniform computations are employed then to make the system *sustainable* the government would have to fix a probable size (an upper bound) of the number of employees in all the companies. Also, the size of all ciphertexts ever generated would scale with that upper bound even if the number of employees in some companies, at the time of encryption, are much smaller than that upper bound. This motivates us to consider the following problem.

Open problem *Can we construct an* FE *scheme for* AWS*in some uniform computational model capable of handling public/private attributes of arbitrary length?*

Our results This work resolves the above open problem. For the *first* time in the literature, we formally define and construct an FE scheme for *unbounded* AWS (UAWS) functionality where the setup only depends on the security parameter of the system and the weight functions are modeled as Turing machines. The proposed FE scheme supports both public and private attributes of *arbitrary* lengths. In particular, the public parameters of the system are completely independent of the lengths of attribute pairs. Moreover, the ciphertext size is *compact* meaning that it does not grow with the number of occurrences of a specific attribute in the weight functions which are represented as Logspace Turing machines. As a special case, we also obtain a FE scheme for attribute-weighted sums where the weight functions are modelled as deterministic finite automata (DFA). The schemes are *adaptively simulation* secure against the release of an unbounded (polynomial) number of secret keys both before and after the challenge ciphertext. As noted in [24, 69], simulation security is the best possible and the most desirable model for FE. Moreover, simulation-based security also captures indistinguishability-based security but the converse does not hold in general.

Our FE for UAWS is proven secure in the standard model based on the symmetric external Diffie–Hellman (SXDH) assumption in the asymmetric prime-order pairing groups. Our main result in the paper is summarized as follows.

Theorem 1 (Informal) *Assuming the* SXDH *assumption holds in asymmetric pairing groups of prime-order, there exists an adaptively simulation secure* FE *scheme for the attribute weighted sum functionality with the weight functions modeled as Logspace Turing machines such that the lengths of public and private attributes are unbounded and can be chosen at the time of encryption, the ciphertexts are compact with respect to the multiple occurrences of attributes in the weight functions.*

Viewing IPFE as a special case of FE for AWS, we also obtain the *first* adaptively *simulation* secure IPFE scheme for *unbounded* length vectors (UIPFE), i.e., the length of the vectors is not fixed in setup. Observe that all prior simulation secure IPFE [8, 10, 38, 76] could only support *bounded* length vectors, i.e., the lengths must be fixed in the setup. On the other hand, the only known construction of UIPFE [71] is proven secure in the *indistinguishability-based* model.

The proposed FE construction is semi-generic and extends the frameworks of the works of Lin and Luo [62] and Datta and Pal [38]. Lin and Luo [62] develop an adaptively secure attribute-based encryption (ABE) scheme for Logspace Turing machines proven secure in the indistinguishability-based model. Although the input length of their ABE is unbounded, but an ABE is an "*all-or-nothing*" type primitive which fully discloses the message to a secret key generated for accepting policies. Further, the ABE of [62] is only payload hiding secure meaning that the ciphertexts themselves can leak sensitive information about the associated attributes. In contrast, our FE for UAWS provides more fine grained encryption methodologies where the ciphertexts reveal nothing about the private part of associated attributes but their weighted sums. Our FE construction depends on two building blocks, an *arithmetic key garbling scheme* (AKGS) for Logspace Turing machines which is an information-theoretic tool and a function hiding (bounded) slotted IPFE scheme which is a computational primitive. An important motivation of [62] is to achieve compact ciphertexts for ABEs. In other words, they get rid of the so-called *one-use restriction* from prior adaptively secure ABEs [19, 30, 31, 57–59, 67, 68, 75] by replacing the core information-theoretic step with the computational primitive of function hiding slotted IPFE. The FE of [38] is able to accomplish this property for non-uniform computations by developing a three-slot encryption technique. Specifically, three slots are utilized to simulate the label functions obtained from the underlying AKGS garbling for pre-ciphertext secret keys. Note that, the three-slot encryption technique is an extension of dual system encryption methodologies [57, 58, 73]. In this work, we extend their frameworks [38, 62] to avoid the one-use restriction in the case of FE for UAWS that computes weights via Logspace Turing machines. It is non-trivial to implement such three-slot techniques in the uniform model. The main reason behind this fact is that in case of ABPs [38] the garbling randomness can be sampled knowing the size of ABPs, and hence the garbling algorithm is possible to run while generating secret keys. However, in the case of AKGS for Logspace Turing machines, the garbling randomness depends on the size of the Turing machine as well as its input lengths. Consequently, it is not possible to execute the garbling in the key generation or encryption algorithms as the information about the garbling randomness is distributed between these two algorithms. We tackle this by developing a more advanced three-slot encryption technique with *distributed randomness* which enables us to carry out such a sophisticated procedure for Logspace Turing machines.

Our FE for UAWS is a one-slot scheme. This means one pair of public–private attribute can be processed in a single encryption. An unbounded-slot FE for UAWS [8] enables us to encrypt unbounded many such pairs in a single encryption. Abdalla et al. [8] devise a generic transformation for bootstrapping from one-slot to unbounded-slot scheme. However, this transformation only works if the underlying one-slot scheme is semi-adaptively secure [38]. Thus, if we restrict our scheme to semi-adaptive security then using such transformations [8, 38] our one-slot FE scheme can be bootstrapped to support unbounded slots.

Current vs. preliminary versions A preliminary version [39] of this work has appeared in Asiacrypt 2022. This paper includes a significant and considerable amount of technical contributions compared to the preliminary version [39]. The previous version contains only the constructions of our single key, single ciphertext secure one-slot FE scheme and the one-slot FE scheme for Logspace without providing any formal treatment to the security analysis of these protocols. The preliminary version presents a very high level idea about the security analysis. Therefore, most of our technical contributions are not formalized in that version. We emphasize that representing and formalizing a proper sequence of hybrid experiments for the security analysis are crucial for understanding the technical challenges and their solutions which we provide in the current version. Especially, we not only describe a proof sketch (for each security analysis) but also depict the hybrid experiment in several tables (see Sects. 5 and 6) that clearly gives a concrete idea about the steps to prove the adaptive simulation security of our schemes. For example, the three-slot reduction mechanism devised in this paper for handling the pre-ciphertext keys of the one-slot FE scheme for Logspace Turing machines are described in Tables 14, 15 and 16. Moreover, in Sect. 7 of the current version, we build a simpler FE scheme for attribute-weighted sums for deterministic finite automata or DFA. Note that, weight functions realized by DFA captures many real-life applications involving computation on unbounded data (or attributes) such as network logging, tax returns and virus scanners. Hence, our FE for DFA becomes more effective compared to the FE for Logspace Turing machines for such potential applications.

Organization We discuss a detailed technical overview of our results in Sect. 2. We provide useful notations, related definitions, and complexity assumptions in Sect. 3. We give a description of AKGS construction for evaluating Turing machines via a sequence of matrix multiplications in Sect. 4. Our construction of a single key and single ciphertext secure FE scheme for UAWS can be found in Sect. 5. We provide the complete security analysis of the scheme in Sect. 5.2. Next, we build our full fledge 1-slot FE scheme for UAWS and prove its security in Sect. 6. We present our FE scheme for attribute-weighted sums for DFA in Sect. 7.

2 Technical overview

We now present an overview of our techniques for achieving an FE scheme for AWS functionality which supports the uniform model of computations. We consider prime-order bilinear pairing groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ with a generator $g_T = e(g_1, g_2)$ of $\mathbb{G}_T$ and denote $[\![a]\!]_i$ by an element $g_i^a \in \mathbb{G}_i$ for $i \in \{1, 2, T\}$. For any vector $\boldsymbol{z}$, the k-th entry is denoted by $\boldsymbol{z}[k]$ and $[n]$ denotes the set $\{1, \ldots, n\}$.

The unbounded AWS functionality In this work, we consider an unbounded FE scheme for the AWS functionality for Logspace Turing machines (or the class of L), in shorthand it is written as $\mathsf{UAWS}^{\mathsf{L}}$. More specifically, the setup only takes as input the security parameter of the system and is independent of any other parameter, e.g., the lengths of the public and private attributes. $\mathsf{UAWS}^{\mathsf{L}}$ generates secret keys $\mathsf{SK}_{(\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}})}$ for a tuple of Turing machines denoted by $\boldsymbol{M} = \{M_k\}_{k\in\mathcal{I}_{\boldsymbol{M}}}$ such that the index set $\mathcal{I}_{\boldsymbol{M}}$ contains any *arbitrary* number of Turing machines $M_k \in \mathsf{L}$. The ciphertexts are computed for a pair of public–private attributes $(\boldsymbol{x}, \boldsymbol{z})$ whose lengths are *arbitrary* and are decided at the time of encryption. Precisely, the public attribute $\boldsymbol{x}$ of length N comes with a polynomial time bound $T = \mathsf{poly}(N)$ and a logarithmic space bound S, and the private attribute $\boldsymbol{z}$ is an integer vector of length n. At the time of decryption, if $\mathcal{I}_{\boldsymbol{M}} \subseteq [n]$ then it reveals an integer value $\sum_{k\in\mathcal{I}_{\boldsymbol{M}}} M_k(\boldsymbol{x})\boldsymbol{z}[k]$. Since $M_k(\boldsymbol{x})$ is binary, we observe that the summation selects and adds the entries of $\boldsymbol{z}$ for which the corresponding Turing machine accepts the public attribute $\boldsymbol{x}$. On the other hand, if $\mathcal{I}_{\boldsymbol{M}}]$ is not contained in $[n]$ then the decryption cannot recover a meaningful information. An appealing feature of the functionality is that the secret key $\mathsf{SK}_{(\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}})}$ can decrypt ciphertexts of unbounded length attributes in unbounded time/(logarithmic) space bounds. In contrast, existing FE for AWSs [8, 38] are designed to handle non-uniform computations that can only handle attributes of bounded lengths and the public parameters grows linearly with the lengths. Next, we describe the formulation of Turing machines in L considered in $\mathsf{UAWS}^{\mathsf{L}}$.

Turing machines formulation We introduce the notations for Logspace Turing machines (TM) over binary alphabets. A Turing machine $M = (Q, \boldsymbol{y}_{\mathsf{acc}}, \delta)$ consists of Q states with the initial state being 1 and a characteristic vector $\boldsymbol{y}_{\mathsf{acc}} \in \{0,1\}^Q$ of accepting states and a transition function δ. When an input $(\boldsymbol{x}, N, T, S)$ with length N and time, space bounds T, S is provided, the computation of $M|_{N,T,S}(\boldsymbol{x})$ is performed in T steps passing through *configurations* $(\boldsymbol{x}, (i, j, \boldsymbol{W}, q))$ where $i \in [N]$ is the input tape pointer, $j \in [S]$ is the work tape pointer, $\boldsymbol{W} \in \{0,1\}^S$ the content of work tape, and $q \in [Q]$ the state under consideration. The initial *internal* configuration is $(1, 1, \mathbf{0}_S, 1)$ and the transition function δ determines whether, on input $\boldsymbol{x}$, it is possible to move from one internal configuration $(i, j, \boldsymbol{W}, q)$ to the next $((i', j', \boldsymbol{W}', q'))$, namely if $\delta(q, \boldsymbol{x}[i], \boldsymbol{W}[j]) = (q', w', \Delta i, \Delta j)$. In other words, the transition function δ on input state q, an input bit $\boldsymbol{x}[i]$ and an work tape bit $\boldsymbol{W}[j]$, outputs the next state q', the new bit w' overwriting $w = \boldsymbol{W}[j]$ by $w' = \boldsymbol{W}'[j]$ (keeping $\boldsymbol{W}[j''] = \boldsymbol{W}'[j'']$ for all $j \neq j''$), and the directions $\Delta i, \Delta j \in \{0, \pm 1\}$ to move the input and work tape pointers.

Our construction of adaptively simulation secure $\mathsf{UAWS}^{\mathsf{L}}$ depends on two building blocks: AKGS for Logspace Turing machines, an information-theoretic tool and slotted IPFE, a computationally secure tool. We only need a *bounded* slotted IPFE, meaning that the length of vectors of the slotted IPFE is fixed in the setup, and we only require the primitive to satisfy adaptive indistinguishability based security. Hence, our work shows how to (semi-)generically bootstrap a bounded IPFE to an unbounded FE scheme beyond the inner product functionality. Before going to describe the $\mathsf{UAWS}^{\mathsf{L}}$, we briefly discuss these two building blocks.

AKGS for Logspace Turing machines In [62], the authors present an ABE scheme for Logspace Turing machines by constructing an efficient AKGS for sequence of matrix multiplications over $\mathbb{Z}_p$. Thus, their core idea was to represent a Turing machine computation through a sequence of matrix multiplications. An internal configuration $(i, j, \boldsymbol{W}, q)$ is represented as a basis vector $\boldsymbol{e}_{(i,j,\boldsymbol{W},q)}$ of dimension $NS2^SQ$ with a single 1 at the position $(i, j, \boldsymbol{W}, q)$. We define a *transition matrix* given by

$$\mathbf{M}(\boldsymbol{x})[(i, j, \boldsymbol{W}, q), (i', j', \boldsymbol{W}', q')] = \begin{cases} 1, & \text{if } \delta(q, \boldsymbol{x}[i], \boldsymbol{W}[j]) = (q', \boldsymbol{W}'[j], i' - i, j' - j) \\ & \quad\text{and } \boldsymbol{W}'[j''] = \boldsymbol{W}[j''] \text{ for all } j'' \neq j; \\ 0, & \text{otherwise;} \end{cases}$$

such that $\boldsymbol{e}^{\top}_{(i,j,\boldsymbol{W},q)}\mathbf{M}(\boldsymbol{x}) = \boldsymbol{e}^{\top}_{(i',j',\boldsymbol{W}',q')}$. This holds because the $((i,j,\boldsymbol{W},q),(i',j',\boldsymbol{W}',q'))$-th entry of $\mathbf{M}(\boldsymbol{x})$ is 1 if and only if there is a valid transition from $(q,\boldsymbol{x}[i],\boldsymbol{W}[j])$ to $(q',\boldsymbol{W}'[j],i'-i,j'-j)$. Therefore, one can write the Turing machine computation by right multiplying the matrix $\mathbf{M}(\boldsymbol{x})$ for T times with the initial configuration $\boldsymbol{e}^{\top}_{(1,1,\mathbf{0}_S,1)}$ to reach of one of the final configurations $\mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \boldsymbol{y}_{\mathsf{acc}}$. In other words, the function $M|_{N,T,S}(\boldsymbol{x})$ is written as

$$M|_{N,T,S}(\boldsymbol{x}) = \boldsymbol{e}^{\top}_{(1,1,\mathbf{0}_S,1)}(\mathbf{M}_{N,S}(\boldsymbol{x}))^T(\mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \boldsymbol{y}_{\mathsf{acc}}) \tag{1}$$

Thus, [62] constructs an AKGS for the sequence of matrix multiplications as in Eq. (1). Their AKGS is inspired from the randomized encoding scheme of [18] and homomorphic evaluation procedure of [22]. Given the function $M|_{N,T,S}$ over $\mathbb{Z}_p$ and two secrets z,β, the garbling procedure computes the label functions

$$\begin{aligned} L_{\mathsf{init}}(\boldsymbol{x}) &= \beta + \boldsymbol{e}^{\top}_{(1,1,\mathbf{0}_S,1)}\boldsymbol{r}_0, \\ \text{for } t\in[T]: \quad (L_{t,\theta})_\theta(\boldsymbol{x}) &= -\boldsymbol{r}_{t-1} + \mathbf{M}_{N,S}(\boldsymbol{x})\boldsymbol{r}_t, \\ (L_{T+1,\theta})_\theta(z) &= -\boldsymbol{r}_T + z\mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \boldsymbol{y}_{\mathsf{acc}}. \end{aligned}$$

and outputs the coefficients of these label functions $\ell_{\mathsf{init}}, \boldsymbol{\ell}_t = (\boldsymbol{\ell}_{t,\theta})_\theta$ where $\theta = (i,j,\boldsymbol{W},q)$ and $\boldsymbol{r}_t \leftarrow \mathbb{Z}_p^{[N]\times[S]\times\{0,1\}^S\times[Q]}$. To compute the functional value for an input $\boldsymbol{x}$, the evaluation procedure adds ℓ_{init} with a telescoping sum $\boldsymbol{e}^{\top}_{(1,1,\mathbf{0}_S,1)} \cdot \sum_{t=1}^{T}(\mathbf{M}_{N,S}(\boldsymbol{x}))^{t-1}\boldsymbol{\ell}_t$ and outputs $zM|_{N,T,S}(\boldsymbol{x}) + \beta$. More precisely, it uses the fact that

$$\begin{aligned} &\boldsymbol{e}^{\top}_{i_{t+1},j_{t+1},\boldsymbol{W}_{t+1},q_{t+1}}\boldsymbol{r}_{t+1} \\ &= \boldsymbol{e}^{\top}_{i_t,j_t,\boldsymbol{W}_t,q_t}\boldsymbol{r}_t + \boldsymbol{e}^{\top}_{i_t,j_t,\boldsymbol{W}_t,q_t}\underbrace{(-\boldsymbol{r}_t + \mathbf{M}(\boldsymbol{x})\boldsymbol{r}_{t+1})}_{\boldsymbol{\ell}_{t+1}} \end{aligned}$$

A crucial and essential property that the AKGS have is the *linearity* of evaluation procedure, meaning that the procedure is linear in the label function values ℓs and, hence can be performed even if the ℓs are available in the exponent of a group. Lin and Luo identify two important security notions of AKGS, jointly called *piecewise security*. Firstly, ℓ_{init} can be reversely sampled given a functional value and all other label values, which is known as the *reverse sampleability*. Secondly, ℓ_t is random with respect to the subsequent label functions $L_{t',\theta}$ for all $t' > t$ and z, which is called the *marginal randomness*.

Function hiding slotted IPFE A normal IPFE computes inner product between two vectors $\boldsymbol{v}$ and $\boldsymbol{u}$ using a secret key $\mathsf{IPFE.SK}_{\boldsymbol{v}}$ and a ciphertext $\mathsf{IPFE.CT}_{\boldsymbol{u}}$. The IPFE is said to satisfy indistinguishability-based security if an adversary having received many functional secret keys $\{\mathsf{IPFE.SK}_{\boldsymbol{v}}\}$ remains incapable to extract any information about the message vector $\boldsymbol{u}$ except the inner products $\{\boldsymbol{v}\cdot\boldsymbol{u}\}$. It is easy to observe that if encryption is done publicly then no security can be ensured about $\boldsymbol{v}$ from the secret key $\mathsf{IPFE.SK}_{\boldsymbol{v}}$ [36] due to the linear functionality. However, if the encryption algorithm is private then $\mathsf{IPFE.SK}_{\boldsymbol{v}}$ can be produced in a fashion to hide sensitive information about $\boldsymbol{v}$. This is termed as *function hiding* security notion for private key IPFE. Slotted IPFE [64] is a hybrid of public and private IPFE where vectors are divided into public and private slots, and function hiding is only guaranteed for the entries in the private slots. Further, Slotted IPFEs of [62, 64] generate secret keys and ciphertexts even when the vectors are given in the exponent of source groups whereas decryption recovers the inner product in the target group.

2.1 From *all-or-nothing* to *functional* encryption

We are all set to describe our approach to extend the framework of [62] from *all-or-nothing* to *functional* encryption for the uniform model of computations. In a previous work of Datta and Pal [38], an adaptively secure FE for AWS functionality was built for a non-uniform model of computation, ABPs to be precise. Their idea was to garble a function $f_k(\boldsymbol{x})\boldsymbol{z}[k] + \beta_k$ during key generation (keeping $\boldsymbol{z}[k]$ and $\boldsymbol{x}$ as variables) and compute IPFE secret keys to encode the m labels, and a ciphertext associated to a tuple $(\boldsymbol{x}, \boldsymbol{z})$ consists of a collection of IPFE ciphertexts which encode the attributes:

$$\begin{aligned}
\mathsf{SK}_f &= \{\mathsf{IPFE.SK}_{\boldsymbol{v}_{k,t<m}}, \widetilde{\mathsf{IPFE.SK}}_{\widetilde{\boldsymbol{v}}_{k,m}}\}_{k,m} : \\
&\quad \boldsymbol{v}_{k,t<m} = \boldsymbol{\ell}_{k,t}, \widetilde{\boldsymbol{v}}_{k,m} = \boldsymbol{\ell}_{k,m} \text{ where} \\
(\boldsymbol{\ell}_{k,t})_t &\leftarrow \mathsf{Garble}(f_k(\boldsymbol{x})\boldsymbol{z}[k] + \beta_k) \text{ s.t. } \textstyle\sum_k \beta_k = 0 \\
\mathsf{CT}_{\boldsymbol{x}} &= (\mathsf{IPFE.CT}_{\boldsymbol{u}}, \{\widetilde{\mathsf{IPFE.CT}}_{\widetilde{\boldsymbol{u}}_k}\}_k) : \\
\boldsymbol{u} &= (1, \boldsymbol{x}), \ \widetilde{\boldsymbol{u}}_k = (1, \boldsymbol{z}[k])
\end{aligned}$$

Therefore, using the inner product functionality, decryption computes the actual label values with $\boldsymbol{x}, \boldsymbol{z}[k]$ as inputs and recovers $f_k(\boldsymbol{x})\boldsymbol{z}[k]+\beta_k$ for each k, and hence finally $\sum_k f_k(\boldsymbol{x})\boldsymbol{z}[k]$. However, this approach fails to build $\mathsf{UAWS}^{\mathsf{L}}$ because we can not execute the AKGS garbling for the function $M_k|_{N,T,S}(\boldsymbol{x})\boldsymbol{z}[k] + \beta_k$ at the time of generating keys. More specifically, the garbling randomness depends on parameters N, T, S, n that are unknown to the key generator. Note that, in contrast to the ABE of [62] where $\boldsymbol{z}$ can be viewed as a payload (hence $n = 1$), the UAWS functionality has an additional parameter n (length of $\boldsymbol{z}$) the value of which is chosen at the time of encryption. Moreover, the compactness of $\mathsf{UAWS}^{\mathsf{L}}$ necessitates the secret key size $|\mathsf{SK}_{(\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}})}| = O(|\mathcal{I}_{\boldsymbol{M}}|Q)$ to be linear in the number of states Q and the ciphertext size $|\mathsf{CT}_{(\boldsymbol{x},T,S)}| = O(nTNS2^S)$ be linear in $TNS2^S$.

The obstacle is circumvented by the randomness distribution technique used in [62]. Instead of computing the AKGS garblings in key generation or encryption phase, the label values are produced by a joint effort of both the secret key and ciphertext. To do so, the garbling is executed under the hood of IPFE using pseudorandomness, instead of true randomness. That is, some part of the garbling randomness is sampled in key generation whereas the rest is sampled in encryption. More specifically, every true random value $\boldsymbol{r}_t[(i, j, \boldsymbol{W}, q)]$ is written as a product $\boldsymbol{r}_{\boldsymbol{x}}[(t, i, j, \boldsymbol{W})]\boldsymbol{r}_{k,f}[q]$ where $\boldsymbol{r}_{\boldsymbol{x}}[(t, i, j, \boldsymbol{W})]$ is used in the ciphertext and $\boldsymbol{r}_{k,f}[q]$ is utilized to encode the transition blocks of M_k in the secret key. To enable this, the transition matrix associated to M_k is represented as follows:

$$\begin{aligned}
&\mathbf{M}(\boldsymbol{x})[(i, j, \boldsymbol{W}, q), (i', j', \boldsymbol{W}', q')] \\
&= \delta^{(?)}((i, j, \boldsymbol{W}, q), (i', j', \boldsymbol{W}', q')) \times \mathbf{M}_{\boldsymbol{x}[i], \boldsymbol{W}[j], \boldsymbol{W}'[j], i'-i, j'-j}[q, q']
\end{aligned}$$

where $\delta^{(?)}((i, j, \boldsymbol{W}, q), (i', j', \boldsymbol{W}', q'))$ is 1 if there is a valid transition from the configuration $(i, j, \boldsymbol{W}, q)$ to $(i', j', \boldsymbol{W}', q')$, otherwise 0. Therefore, every block of $\mathbf{M}(\boldsymbol{x})[(i, j, \boldsymbol{W}, q),$ $(i', j', \boldsymbol{W}', q')]$ is either a $Q \times Q$ zero matrix or a *transition block* that belongs to a small set

$$\mathcal{T} = \{\mathbf{M}_\tau | \ \tau = (x, w, w', \Delta i, \Delta j) \in \{0, 1\}^3 \times \{0, \pm 1\}^2\}$$

The $(i, j, \boldsymbol{W}, q)$-th *block row* $\mathbf{M}_\tau = \mathbf{M}_{x,w,w',\Delta i,\Delta j}$ appears at $\mathbf{M}_{N,S}(\boldsymbol{x})[(i, j, \boldsymbol{W}, \lrcorner),$ $(i', j', \boldsymbol{W}', \lrcorner)]$ if $x = \boldsymbol{x}[i], w = \boldsymbol{W}[j], \Delta i = i' - i, \Delta j = j' - j$, and $\boldsymbol{W}'$ is $\boldsymbol{W}$ with j-th entry changed to w'. Thus, every label $\boldsymbol{\ell}_{k,t}[\mathrm{i}, q]$ with $\mathrm{i} = (i, j, \boldsymbol{W})$ can be *decomposed* as inner product $\boldsymbol{v}_{k,q} \cdot \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$. More precisely,

$$\begin{aligned}
&\boldsymbol{\ell}_{k,t}[\mathrm{i}, q] \\
&= -\boldsymbol{r}_{t-1}[\mathrm{i}, q] + \mathbf{M}_{k,N,S}(\boldsymbol{x})[(\mathrm{i}, q), (\llcorner, \llcorner, \llcorner, \llcorner)]\boldsymbol{r}_t \\
&= -\boldsymbol{r}_{t-1}[\mathrm{i}, q] + \sum_{w', \Delta i, \Delta j} (\mathbf{M}_{k,\boldsymbol{x}[i],\boldsymbol{W}[j],w',\Delta i,\Delta j}\boldsymbol{r}_t[\mathrm{i}', \llcorner])[q] \\
&\qquad\qquad\qquad\qquad\qquad (\mathrm{i}' = (i + \Delta i, j + \Delta j, \boldsymbol{W}')) \\
&= \boldsymbol{r}_{\boldsymbol{x}}[t-1, \mathrm{i}]\boldsymbol{r}_{k,f}[q] + \sum_{w', \Delta i, \Delta j} \boldsymbol{r}_{\boldsymbol{x}}[t, \mathrm{i}'](\mathbf{M}_{k,\boldsymbol{x}[i],\boldsymbol{W}[j],w',\Delta i,\Delta j}\boldsymbol{r}_{k,f})[q] \\
&= \boldsymbol{r}_{\boldsymbol{x}}[t-1, \mathrm{i}]\boldsymbol{r}_{k,f}[q] + \sum_{w', \Delta i, \Delta j} \boldsymbol{r}_{\boldsymbol{x}}[t, \mathrm{i}'](\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q] = \boldsymbol{v}_{k,q} \cdot \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}
\end{aligned}$$

so that one can set the vectors

$$\begin{aligned}
\boldsymbol{v}_{k,q} &= (\ -\boldsymbol{r}_{k,f}[q],\ \ 0,\ (\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q]\ \ \|\ \mathbf{0}\ ), \\
\boldsymbol{u}_{t,\mathrm{i}} &= (\ \boldsymbol{r}_{\boldsymbol{x}}[t-1, \mathrm{i}],\ 0,\ \ \ c_\tau(\boldsymbol{x}; \boldsymbol{r}_{\boldsymbol{x}})\ \ \ \|\ \mathbf{0}\ )
\end{aligned}$$

where $c_\tau(\boldsymbol{x}; \boldsymbol{r}_{\boldsymbol{x}})$ (a shorthand of the notation $c_\tau(\boldsymbol{x}, t, i, j, \boldsymbol{W}; \boldsymbol{r}_{\boldsymbol{x}})$ [62]) is given by

$$c_\tau(\boldsymbol{x}; \boldsymbol{r}_{\boldsymbol{x}}) = \begin{cases} \boldsymbol{r}_{\boldsymbol{x}}[t, \mathrm{i}'], & \text{if } x = \boldsymbol{x}[i], w = \boldsymbol{W}[j]; \\ 0, & \text{otherwise.} \end{cases}$$

Similarly, the other labels can be decomposed: $\ell_{k,\mathsf{init}} = (\boldsymbol{r}_{k,f}[1], \beta_k, 0) \cdot (\boldsymbol{r}_{\boldsymbol{x}}[(0, 1, 1, \mathbf{0}_S)], 1, 0) = \beta_k + \boldsymbol{e}^\top_{(1,1,\mathbf{0}_S,1)}\boldsymbol{r}_0$ and $\boldsymbol{\ell}_{k,T+1}[(\mathrm{i}, q)] = \widetilde{\boldsymbol{v}}_{k,q} \cdot \widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}} = -\boldsymbol{r}_T[(\mathrm{i}, q)] + z[k]\boldsymbol{y}_{k,\mathsf{acc}}[q]$ where

$$\begin{aligned}
\widetilde{\boldsymbol{v}}_{k,q} &= (\ -\boldsymbol{r}_{k,f}[q],\ \boldsymbol{y}_{k,\mathsf{acc}}[q]\ \ \|\ \mathbf{0}\ ), \\
\widetilde{\boldsymbol{u}}_{T+1,\mathrm{i}} &= (\ \ \boldsymbol{r}_{\boldsymbol{x}}[T, \mathrm{i}],\ \ \ \ z[k]\ \ \ \ \|\ \mathbf{0}\ )
\end{aligned}$$

A first attempt Armed with this, we now present the first candidate $\mathsf{UAWS}^{\mathsf{L}}$ construction in the secret key setting which supports a single key. We consider two independent master keys imsk and $\widetilde{\mathsf{imsk}}$ of IPFE. For simplicity, we assume the length of the private attribute z is the same as the number of Turing machines present in $\boldsymbol{M} = (M_k)_{k \in \mathcal{I}_{\boldsymbol{M}}}$, i.e., $n = |\mathcal{I}_{\boldsymbol{M}}|$. We also assume that each Turing machine in the secret key shares the same set of states.

$$\begin{aligned}
\mathsf{SK}_{\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}}} &= \{\mathsf{IPFE.SK}_{\boldsymbol{v}_{k,\mathsf{init}}}, \mathsf{IPFE.SK}_{\boldsymbol{v}_{k,q}}, \widetilde{\mathsf{IPFE.SK}}_{\widetilde{\boldsymbol{v}}_{k,q}}\}_{k \in \mathcal{I}_{\boldsymbol{M}}} : \\
[\![\boldsymbol{v}_{k,\mathsf{init}}]\!]_2 &= [\![(\ -\boldsymbol{r}_{k,f}[1],\ \ \ \beta_k,\ \ \ \ \ 0,\ \ \ \ \|\ \mathbf{0}\ )]\!]_2, \\
[\![\boldsymbol{v}_{k,q}]\!]_2 &= [\![(\ -\boldsymbol{r}_{k,f}[q],\ \ \ 0,\ \ (\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q]\ \|\ \mathbf{0}\ )]\!]_2, \\
[\![\widetilde{\boldsymbol{v}}_{k,q}]\!]_2 &= [\![(\ -\boldsymbol{r}_{k,f}[q],\ \boldsymbol{y}_{k,\mathsf{acc}}[q]\ \ \ \ \ \ \|\ \mathbf{0}\ )]\!]_2
\end{aligned}$$

$$\begin{aligned}
\mathsf{CT}_{\boldsymbol{x}} &= (\mathsf{IPFE.CT}_{\boldsymbol{u}_{\mathsf{init}}}, \mathsf{IPFE.CT}_{\boldsymbol{u}}, \{\widetilde{\mathsf{IPFE.CT}}_{\widetilde{\boldsymbol{u}}_k}\}_k) : \\
[\![\boldsymbol{u}_{\mathsf{init}}]\!]_1 &= [\![(\ \boldsymbol{r}_{\boldsymbol{x}}[(0, 1, 1, \mathbf{0}_S)],\ 1,\ \ \ 0,\ \ \ \|\ \mathbf{0}\ )]\!]_1, \\
[\![\boldsymbol{u}_{t<T,\mathrm{i}}]\!]_1 &= [\![(\ \ \boldsymbol{r}_{\boldsymbol{x}}[t-1, \mathrm{i}],\ \ \ 0,\ c_\tau(\boldsymbol{x}; \boldsymbol{r}_{\boldsymbol{x}})\ \|\ \mathbf{0}\ )]\!]_1, \\
[\![\widetilde{\boldsymbol{u}}_{k,T+1,\mathrm{i}}]\!]_1 &= [\![(\ \ \ \boldsymbol{r}_{\boldsymbol{x}}[T, \mathrm{i}],\ \ \ \ z[k]\ \ \ \ \|\ \mathbf{0}\ )]\!]_1
\end{aligned}$$

Observe that the inner products between the ciphertext and secret key vectors yield the label values $[\![\ell_{k,\mathsf{init}}]\!]_{\mathrm{T}}$, $[\![\boldsymbol{\ell}_{k,t}]\!]_{\mathrm{T}} = [\![(\boldsymbol{\ell}_{k,t,\theta})_\theta]\!]_{\mathrm{T}}$ for $\theta = (i, j, \boldsymbol{W}, q)$. Now, the evaluation procedure of AKGS is applied to obtain the partial values $[\![z[k]M_k|_{N,T,S}(\boldsymbol{x}) + \beta_k]\!]_{\mathrm{T}}$. Combining all this values gives the required attribute weighted sum $\sum_k M_k|_{N,T,S}(\boldsymbol{x})z[k]$ Since $\sum_k \beta_k = 0$.

However, this scheme is not *fully* unbounded, in particular, the setup needs to know the length of the private attribute. To realise this, let us try to prove the security of the scheme. The main idea of the proof would be to make all the label values $(\boldsymbol{\ell}_{k,t,\theta})_\theta$ truly random and simulated except the initial labels $\ell_{k,\mathsf{init}}$ so that one can reversely sample $\ell_{k,\mathsf{init}}$ hardcoded with a desired functional value. Suppose, for instance, the single secret key is queried before the challenge ciphertext. In this case, the honest label values are first hardwired in the ciphertext vectors and then the labels are transformed into their simulated version. This is because the ciphertext vectors are computed after the secret key. So, the first step is to hardwire the initial label values $\ell_{k,\mathsf{init}}$ into the ciphertext vector $\boldsymbol{u}_{\mathsf{init}}$, which indicates that the length of $\boldsymbol{u}_{\mathsf{init}}$ must grow with respect to the number of $\ell_{k,\mathsf{init}}$'s. The same situation arises while simulating the other label values through $\boldsymbol{u}_{t,\mathfrak{i}}$. In other words, we need to know the size of $\mathcal{I}_{\boldsymbol{M}}$ or the length of $\boldsymbol{z}$ in setup, which is against our desired functionality.

To tackle this, we increase the number of $\boldsymbol{u}_{\mathsf{init}}$ and $\boldsymbol{u}_{t<T,\mathfrak{i}}$ in the above system. More specifically, each of these vectors are now computed for all $k \in [n]$, just like $\widetilde{\boldsymbol{u}}_{k,T+1,\mathfrak{i}}$. Although this fixes the requirement of unboundedness of the system, there is another issue related to the security that must be solved. Note that, in the current structure, there is a possibility of *mix-and-match* attacks since, for example, $\widetilde{\boldsymbol{u}}_{k_1,T+1,\mathfrak{i}}$ can be paired with $\widetilde{\boldsymbol{v}}_{k_2,q}$ and this results in some *unwanted* attribute weighted sum of the form $\sum_{k\neq k_1,k_2} M_k(\boldsymbol{x})\boldsymbol{z}[k] + M_{k_1}(\boldsymbol{x})\boldsymbol{z}[k_2]+M_{k_2}(\boldsymbol{x})\boldsymbol{z}[k_1]$. We employ the *index encoding* technique used in previous works achieving unbounded ABE or IPFE [68, 71] to overcome the attack. In particular, we add two extra dimensions $\rho_k(-k, 1)$ in the ciphertext and $\pi_k(1, k)$ in the secret key for encoding the index k in each of the vectors of the system. Observe that for each Turing machine M_k an independent randomness π_k is sampled. It ensures that an adversary can only recover the desired attribute weighted sum and whenever vectors from different indices are paired only a garbage value is obtained.

Combining the ideas After combining the above ideas, we describe our $\mathsf{UAWS}^{\mathsf{L}}$ supporting a single key as follows.

$$\mathsf{SK}_{\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}}} = \{\mathsf{IPFE.SK}_{\boldsymbol{v}_{k,\mathsf{init}}}, \mathsf{IPFE.SK}_{\boldsymbol{v}_{k,q}}, \widetilde{\mathsf{IPFE.SK}}_{\widetilde{\boldsymbol{v}}_{k,q}}\}_{k\in\mathcal{I}_{\boldsymbol{M}}}:$$

$$\begin{aligned}
[\![\boldsymbol{v}_{k,\mathsf{init}}]\!]_2 &= [\![(\ \pi_k(1,k),\ -\boldsymbol{r}_{k,f}[1],\quad \beta_k,\qquad 0,\qquad \|\ \boldsymbol{0}\ )]\!]_2,\\
[\![\boldsymbol{v}_{k,q}]\!]_2 &= [\![(\ \pi_k(1,k),\ -\boldsymbol{r}_{k,f}[q],\quad 0,\quad (\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q]\ \|\ \boldsymbol{0}\ )]\!]_2,\\
[\![\widetilde{\boldsymbol{v}}_{k,q}]\!]_2 &= [\![(\ \pi_k(1,k),\ -\boldsymbol{r}_{k,f}[q],\ \boldsymbol{y}_{k,\mathsf{acc}}[q]\qquad \|\ \boldsymbol{0}\ )]\!]_2
\end{aligned}$$

$$\mathsf{CT}_{\boldsymbol{x}} = \{\mathsf{IPFE.CT}_{\boldsymbol{u}_{k,\mathsf{init}}}, \mathsf{IPFE.CT}_{\boldsymbol{u}_{k,t<T,\mathfrak{i}}}, \widetilde{\mathsf{IPFE.CT}}_{\widetilde{\boldsymbol{u}}_{k,T+1,\mathfrak{i}}}\}_k:$$

$$\begin{aligned}
[\![\boldsymbol{u}_{k,\mathsf{init}}]\!]_1 &= [\![(\ \rho_k(-k,1),\ \boldsymbol{r}_{\boldsymbol{x}}[(0,1,1,\boldsymbol{0}_S)],\ 1,\qquad 0,\qquad \|\ \boldsymbol{0}\ )]\!]_1,\\
[\![\boldsymbol{u}_{k,t<T,\mathfrak{i}}]\!]_1 &= [\![(\ \rho_k(-k,1),\quad \boldsymbol{r}_{\boldsymbol{x}}[t-1,\mathfrak{i}],\quad 0,\ c_\tau(\boldsymbol{x};\boldsymbol{r}_{\boldsymbol{x}})\ \|\ \boldsymbol{0}\ )]\!]_1,\\
[\![\widetilde{\boldsymbol{u}}_{k,T+1,\mathfrak{i}}]\!]_1 &= [\![(\ \rho_k(-k,1),\quad \boldsymbol{r}_{\boldsymbol{x}}[T,\mathfrak{i}],\quad \boldsymbol{z}[k]\qquad \|\ \boldsymbol{0}\ )]\!]_1
\end{aligned}$$

Although the above construction satisfies our desired functionality, preserves the compactness of ciphertexts and resists the aforementioned attack, we face multiple challenges in adapting the proof ideas of previous works [38, 62, 71].

Security challenges and solutions Next, we discuss the challenges in proving the adaptive simulation security of the scheme. Firstly, the unbounded IPFE scheme of Tomida and Takashima [71] is proved in the *indistinguishability-based* model whereas we aim to prove simulation security which is much more challenging. The work closer to ours is the FE for AWS of Datta and Pal [38], but it only supports a *non-uniform* model of computation and the inner product functionality is *bounded*. Moreover, since the garbling randomness is distributed in the secret key and ciphertext vectors, we can not adapt their proof techniques [38, 71] in a straightforward manner. Although the ABE scheme of Lin and Luo [62] handles a uniform model of computation, they only consider *all-or-nothing* type encryptions and hence the adversary is allowed to query secret keys which always fail to decrypt the challenge ciphertext. In contrast, we construct a more advanced encryption mechanism which overcomes all the above constraints of prior works, i.e., our $\mathsf{UAWS}^{\mathsf{L}}$ is an adaptively *simulation* secure *functional encryption scheme* that supports *unbounded* inner product functionality with a *uniform* model of computations over the public attributes.

Our proof technique is inspired by that of [38, 62]. One of the core technical challenges is involved in the case where the secret key is queried before the challenge ciphertext. Thus, we focus more on "sk queried before ct" in this overview. As noted above, in the security analysis of [62] the adversary $\mathcal{A}$ is not allowed to decrypt the challenge ciphertext and hence they completely randomize the ciphertext in the final game. However, since we are building a FE scheme any secret key queried by $\mathcal{A}$ should be able to decrypt the challenge ciphertext. For this, we use the pre-image sampleability technique from prior works [37, 38]. In particular, the reduction samples a dummy vector $\boldsymbol{d} \in \mathbb{Z}_p^n$ satisfying $\sum_k M_k|_{N,T,S}(\boldsymbol{x})\boldsymbol{z}[k] = \sum_k M_k|_{N,T,S}(\boldsymbol{x})\boldsymbol{d}[k]$ where $\boldsymbol{M} = (M_k)_k$ is a pre-challenge secret key. To plant the dummy vector into the ciphertext, we first need to make all label values $\{\ell_{k,t,\mathrm{i},q}\}$ truly random depending on the terms $\boldsymbol{r}_{k,f}[q]\boldsymbol{r}_{\boldsymbol{x}}[t-1,\mathrm{i}]$'s and then turn them into their simulated forms, and finally traverse in the reverse path to get back the original form of the ciphertext with $\boldsymbol{d}$ taking place of the private attribute $\boldsymbol{z}$. In order to make all these labels truly random, the honest label values are needed to be hardwired into the ciphertext vectors (since these are computed later) so that we can apply the DDH assumption in $\mathbb{G}_1$ to randomize the term $\boldsymbol{r}_{k,f}[q]\boldsymbol{r}_{\boldsymbol{x}}[t-1,\mathrm{i}]$ (hence the label values). However, this step is much more complicated than in [62] since there are two independent IPFE systems in our construction and $\boldsymbol{r}_{k,f}[q]$ appears in both $\boldsymbol{v}_{k,q}$ and $\widetilde{\boldsymbol{v}}_{k,q}$ (i.e., in both IPFE systems). We design a two-level nested loop running over q and t for relocating $\boldsymbol{r}_{k,f}[q]$ from $\boldsymbol{v}$'s and $\widetilde{\boldsymbol{v}}_{k,q}$ to $\boldsymbol{u}$'s and $\widetilde{\boldsymbol{u}}_{k,T+1,\mathrm{i}}$. To this end, we note that the case of "sk queried after ct" is simpler where we embed the reversely sampled initial label values into the secret key. Before going to discuss the hybrids, we first present the *simulator* of the ideal world.

$$\begin{aligned}
\mathsf{SK}_{\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}}} &= \{\mathsf{IPFE.SK}_{\boldsymbol{v}_{k,\mathsf{init}}}, \mathsf{IPFE.SK}_{\boldsymbol{v}_{k,q}}, \widetilde{\mathsf{IPFE.SK}}_{\widetilde{\boldsymbol{v}}_{k,q}}\}_{k\in\mathcal{I}_{\boldsymbol{M}}} : \text{ (sk queried before ct)}\\
[\![\boldsymbol{v}_{k,\mathsf{init}}]\!]_2 &= [\![(\ \pi_k(1,k),\ -\boldsymbol{r}_{k,f}[1],\quad \beta_k,\qquad\qquad 0 \qquad\qquad \|\ \mathbf{0}\ )]\!]_2,\\
[\![\boldsymbol{v}_{k,q}]\!]_2 &= [\![(\ \pi_k(1,k),\ -\boldsymbol{r}_{k,f}[q],\quad 0,\quad (\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q]\ \|\ \mathbf{0}\ )]\!]_2,\\
[\![\widetilde{\boldsymbol{v}}_{k,q}]\!]_2 &= [\![(\ \pi_k(1,k),\ -\boldsymbol{r}_{k,f}[q],\ \boldsymbol{y}_{k,\mathsf{acc}}[q] \qquad\qquad\qquad \|\ \mathbf{0}\ )]\!]_2
\end{aligned}$$

$$\begin{aligned}
\mathsf{CT}_{\boldsymbol{x}} &= \{\mathsf{IPFE.CT}_{\boldsymbol{u}_{k,\mathsf{init}}}, \mathsf{IPFE.CT}_{\boldsymbol{u}_{k,t<T,\mathrm{i}}}, \widetilde{\mathsf{IPFE.CT}}_{\widetilde{\boldsymbol{u}}_{k,T+1,\mathrm{i}}}\}_k :\\
[\![\boldsymbol{u}_{k,\mathsf{init}}]\!]_1 &= [\![(\ \rho_k(-k,1),\ \boldsymbol{r}_{\boldsymbol{x}}[(0,1,1,\mathbf{0}_S)],\ 1,\qquad 0,\qquad \|\qquad 1,\qquad \mathbf{0}\ )]\!]_1,\\
[\![\boldsymbol{u}_{k,t<T,\mathrm{i}}]\!]_1 &= [\![(\ \rho_k(-k,1),\quad \boldsymbol{r}_{\boldsymbol{x}}[t-1,\mathrm{i}],\quad 0,\ c_\tau(\boldsymbol{x};\boldsymbol{r}_{\boldsymbol{x}})\ \|\quad \boldsymbol{s}_{\boldsymbol{x}}[t,\mathrm{i}],\quad \mathbf{0}\ )]\!]_1,\\
[\![\widetilde{\boldsymbol{u}}_{k,T+1,\mathrm{i}}]\!]_1 &= [\![(\ \rho_k(-k,1),\qquad \boldsymbol{r}_{\boldsymbol{x}}[T,\mathrm{i}],\qquad \boldsymbol{d}[k] \qquad\qquad \|\ \boldsymbol{s}_{\boldsymbol{x}}[T+1,\mathrm{i}],\ \mathbf{0}\ )]\!]_1
\end{aligned}$$

$$\begin{aligned}
\mathsf{SK}_{\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}}} &= \{\mathsf{IPFE.SK}_{\boldsymbol{v}_{k,\mathsf{init}}}, \mathsf{IPFE.SK}_{\boldsymbol{v}_{k,q}}, \widetilde{\mathsf{IPFE.SK}}_{\widetilde{\boldsymbol{v}}_{k,q}}\}_{k\in\mathcal{I}_{\boldsymbol{M}}} : \text{ (sk queried after ct)}\\
[\![\boldsymbol{v}_{k,\mathsf{init}}]\!]_2 &= [\![(\ \pi_k(1,k),\ 0,\ 0,\ 0\ \|\ \ell_{k,\mathsf{init}},\ \mathbf{0}\ )]\!]_2,\\
[\![\boldsymbol{v}_{k,q}]\!]_2 &= [\![(\ \pi_k(1,k),\ 0,\ 0,\ 0\ \|\ \boldsymbol{s}_{k,f}[q],\ \mathbf{0}\ )]\!]_2,\\
[\![\widetilde{\boldsymbol{v}}_{k,q}]\!]_2 &= [\![(\ \pi_k(1,k),\ 0,\ 0\quad \|\ \boldsymbol{s}_{k,f}[q],\ \mathbf{0}\ )]\!]_2
\end{aligned}$$

where $\ell_{k,\mathsf{init}} \leftarrow \mathsf{RevSamp}((M_k, \boldsymbol{x}, M_k[\boldsymbol{x}]\boldsymbol{z}[k] + \beta_k, \{\ell_{k,t,\mathrm{i},q}\})$ s.t.
$\sum_{k\in\mathcal{I}_{\boldsymbol{M}}} \beta_k = 0$ if $\mathcal{I}_{\boldsymbol{M}} \subseteq [n]$; otherwise $\beta_k \leftarrow \mathbb{Z}_p$.

Security analysis We use a three-step approach and each step consists of a group of hybrid sequence. At a very high level, we discuss the case of "sk queried before ct". In this overview, for simplicity, we assume that the challenger knows the length of $\boldsymbol{z}$ while it generates the secret key.

First group of hybrids The reduction starts with the real scheme. In the first step, the label function $\ell_{k,\mathsf{init}}$ is reversely sampled with the value $M_k[\boldsymbol{x}]\boldsymbol{z}[k] + \beta_k$ which is hardwired in $\boldsymbol{u}_{k,\mathsf{init}}$.

$$\begin{aligned}
\boldsymbol{v}_{k,\mathsf{init}} &= (\,\cdots,\ \boxed{1},\ \boxed{0},\ 0 \ \|\ 0,\ \mathbf{0}\,),\\
\boldsymbol{v}_{k,q} &= (\,\cdots,\ -\boldsymbol{r}_{k,f}[q],\ 0,\ (\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q] \ \|\ \boxed{s_{k,f}[q]},\ \mathbf{0}\,),\\
\widetilde{\boldsymbol{v}}_{k,q} &= (\,\cdots,\ -\boldsymbol{r}_{k,f}[q],\ \boldsymbol{y}_{k,\mathsf{acc}}[q] \ \|\ 0,\ \mathbf{0}\,)
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{u}_{k,\mathsf{init}} &= (\,\cdots,\ \boxed{\ell_{k,\mathsf{init}}},\ \boxed{0},\ 0, \ \|\ 0,\ \mathbf{0}\,),\\
\boldsymbol{u}_{k,t<T,\mathfrak{i}} &= (\,\cdots,\ \boldsymbol{r}_{\boldsymbol{x}}[t-1,\mathfrak{i}],\ 0,\ c_\tau(\boldsymbol{x};\boldsymbol{r}_{\boldsymbol{x}}) \ \|\ 0,\ \mathbf{0}\,),\\
\widetilde{\boldsymbol{u}}_{k,T+1,\mathfrak{i}} &= (\,\cdots,\ \boldsymbol{r}_{\boldsymbol{x}}[T,\mathfrak{i}],\ z[k] \ \|\ \boxed{\boldsymbol{s}_{\boldsymbol{x}}[T+1,\mathfrak{i}]},\ \mathbf{0}\,)
\end{aligned}$$

where $\ell_{k,\mathsf{init}} \leftarrow \mathsf{RevSamp}((M_k, \boldsymbol{x}, M_k[\boldsymbol{x}]z[k]+\beta_k, \{\ell_{k,t,\mathfrak{i},q}\})$ and $\ell_{k,t,\mathfrak{i},q}$'s are computed honestly. Note that, the secret values $\{\beta_k\}$ are sampled depending on whether the queried key is eligible for decryption. More specifically, if $\mathcal{I}_{\boldsymbol{M}} \subseteq [n]$, then β_k's are sampled as in the original key generation algorithm, i.e., $\sum_k \beta_k = 0$. On the other hand, if $\max\mathcal{I}_{\boldsymbol{M}} > n$ then β_k's are sampled uniformly at random, i.e., they do not necessarily be secret shares of zero. This can be done by the function hiding property of IPFE which ensures that the distributions $\{\{\mathsf{IPFE.SK}_{\boldsymbol{v}_k^{(\mathfrak{b})}}\}_{k\in[n+1,|\mathcal{I}_{\boldsymbol{M}}|]}, \{\mathsf{IPFE.CT}_{\boldsymbol{u}_{k'}}\}_{k'\in[n]}\}$ for $\mathfrak{b} \in \{0,1\}$ are indistinguishable where

$$\begin{aligned}
\boldsymbol{v}_k^{(\mathfrak{b})} &= (\ \ \pi_k,\quad k\cdot\pi_k,\ \mathbf{0},\ \beta_k + \mathfrak{b}\cdot r_k,\ \mathbf{0}\,) \quad \text{for } k \in [n+1, |\mathcal{I}_{\boldsymbol{M}}|], r_k \leftarrow \mathbb{Z}_p\\
\boldsymbol{u}_{k'} &= (\,-k'\cdot\rho_{k'},\ \rho_{k'},\ \mathbf{0},\quad 1,\quad \mathbf{0}\,) \quad \text{for } k' \in [n]
\end{aligned}$$

Thus, the indistinguishability between the group of hybrids can be guaranteed by the piecewise security of AKGS and the function hiding security of IPFE.

Second group of hybrids The second step is a loop. The purpose of the loop is to change all the honest label values $\ell_{k,t,\mathfrak{i},q}$ to simulated ones that take the form $\ell_{k,t,\mathfrak{i},q} = \boldsymbol{s}_{\boldsymbol{x}}[t,\mathfrak{i}]s_{k,f}[q]$ where $\boldsymbol{s}_{\boldsymbol{x}}[t,\mathfrak{i}]$ is hardwired in $\boldsymbol{u}_{k,t,\mathfrak{i}}$ or $\widetilde{\boldsymbol{u}}_{k,T+1,\mathfrak{i}}$ and $s_{k,f}[q]$ is hardwired in $\boldsymbol{v}_{k,q}$ or $\widetilde{v}_{k,q}$.

The whole procedure is executed in via a two-level loop with outer loop running over t and inner loop running over q (both in increasing order). In each iteration of the loop, we move all occurrences of $\boldsymbol{r}_{k,f}[q]$ into the $\boldsymbol{u}$'s in one shot and hardwire the honest labels $\ell_{k,t,\mathfrak{i},q}$ into $\boldsymbol{u}_{k,t,\mathfrak{i}}$ for all $\mathfrak{i}$. Below we present two crucial intermediate hybrids of the loop when $t \leq T$.

$$\begin{aligned}
\boldsymbol{v}_{k,q} &= (\,\cdots,\ -\boxed{\text{✗}\boldsymbol{r}_{k,f}[q]}-\ \|\ \boxed{0},\ \boxed{1},\ \mathbf{0}\,),\\
\widetilde{\boldsymbol{v}}_{k,q} &= (\,\cdots,\ -\boxed{0}-\ \|\ 0,\ \boxed{1},\ \mathbf{0}\,),
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{u}_{k,t<T,\mathfrak{i}} &= (\,\cdots,\ -\boxed{\text{✓}\boldsymbol{r}_{k,f}[q]}-\ \|\ \boxed{\boldsymbol{s}_{\boldsymbol{x}}[t,\mathfrak{i}]},\ \boxed{\begin{array}{l}\text{honest } \ell_{k,t,\mathfrak{i},q}\\ = -\boldsymbol{r}_{\boldsymbol{x}}[t-1,\mathfrak{i}]\boldsymbol{r}_{k,f}[q]+\cdots\end{array}},\ \mathbf{0}\,),\\
\widetilde{\boldsymbol{u}}_{k,T+1,\mathfrak{i}} &= (\,\cdots,\ \boldsymbol{r}_{\boldsymbol{x}}[T,\mathfrak{i}],\ z[k]\ \|\ \boldsymbol{s}_{\boldsymbol{x}}[T+1,\mathfrak{i}],\ \boxed{\begin{array}{l}\text{honest } \ell_{k,T+1,\mathfrak{i},q}\\ = -\boldsymbol{r}_{\boldsymbol{x}}[T,\mathfrak{i}]\boldsymbol{r}_{k,f}[q]+\cdots\end{array}},\ \mathbf{0}\,)
\end{aligned}$$

where $\text{✗}\boldsymbol{r}_{k,f}[q]$ and $\text{✓}\boldsymbol{r}_{k,f}[q]$ indicate the presence of $\boldsymbol{r}_{k,f}[q]$ in their respective positions. The indistinguishability can be argued using the function hiding security of IPFE. Next, by invoking DDH in $\mathbb{G}_1$, we first make $\boldsymbol{r}_{\boldsymbol{x}}[t-1,\mathfrak{i}]\boldsymbol{r}_{k,f}[q]$ truly random for all $\mathfrak{i}$ and then transform the label values into their simulated form $\ell_{k,\mathfrak{i},q} = \boldsymbol{s}_{\boldsymbol{x}}[t,\mathfrak{i}]s_{k,f}[q]$ again by using DDH in $\mathbb{G}_1$ for all $\mathfrak{i}$. We *emphasize* that the labels $\ell_{k,T+1,\mathfrak{i},q}$ are kept as honest and hardwired when the loop runs for $t \leq T$. Finally, the terms $s_{k,f}[q]$ are shifted back to $\boldsymbol{v}_{k,q}$ or $\widetilde{\boldsymbol{v}}_{k,q}$.

$$\begin{aligned}
\boldsymbol{v}_{k,q} &= (\cdots, \boxed{-\boldsymbol{r}_{k,f}[q]}, \quad 0, \quad \boxed{(\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q]} \;\|\; \boxed{s_{k,f}[q]}, \boxed{0}, \mathbf{0}),\\
\widetilde{\boldsymbol{v}}_{k,q} &= (\cdots, \boxed{-\boldsymbol{r}_{k,f}[q]}, \boxed{\boldsymbol{y}_{k,\mathsf{acc}}[q]} \qquad \;\|\; \quad 0, \quad \boxed{0}, \mathbf{0}),
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{u}_{k,t<T,\mathrm{i}} &= (\cdots, \quad -\boxed{0}- \quad \| \quad \boldsymbol{s_x}[t, \mathrm{i}], \quad \boxed{0}, \mathbf{0}),\\
\widetilde{\boldsymbol{u}}_{k,T+1,\mathrm{i}} &= (\cdots, \boldsymbol{r_x}[T, \mathrm{i}], \; z[k] \;\|\; \boldsymbol{s_x}[T+1, \mathrm{i}], \boxed{0}, \mathbf{0}\;)
\end{aligned}$$

After the two-label loop finishes, the reduction run an additional loop over q with t fixed at $T+1$ to make the last few label values $\ell_{k,T+1,\mathrm{i},q}$ simulated. The indistinguishability between the hybrids follows from a similar argument as in the two-level loop.

$$\begin{aligned}
\boldsymbol{v}_{k,q} &= (\cdots, -\boldsymbol{r}_{k,f}[q], \quad 0, \quad (\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q] \;\|\; s_{k,f}[q], \; 0, \mathbf{0}),\\
\widetilde{\boldsymbol{v}}_{k,q} &= (\cdots, -\boldsymbol{r}_{k,f}[q], \; \boldsymbol{y}_{k,\mathsf{acc}}[q] \qquad \;\|\; \boxed{s_{k,f}[q]}, 0, \mathbf{0}),
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{u}_{k,t<T,\mathrm{i}} &= (\cdots, \; -0- \;\|\; \boldsymbol{s_x}[t, \mathrm{i}], \quad 0, \mathbf{0}),\\
\widetilde{\boldsymbol{u}}_{k,T+1,\mathrm{i}} &= (\cdots, -\boxed{0}- \;\|\; \boldsymbol{s_x}[T+1, \mathrm{i}], 0, \mathbf{0}\;)
\end{aligned}$$

Third group of hybrids After all the label values $\ell_{k,t,\mathrm{i},q}$ are simulated, the third step uses a few more hybrids to reversely sample $\ell_{1,\mathsf{init}}$ and $\ell_{k,\mathsf{init}}|_{k>1}$ with the hardcoded values $\boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{z} + \beta_1$ and $\beta_k|_{k>1}$ respectively. This can be achieved through a statistical transformation on $\{\beta_k | \sum_k \beta_k = 0\}$. Finally, we are all set to insert the dummy vector $\boldsymbol{d}$ in place of $\boldsymbol{z}$ keeping $\mathcal{A}$'s view identical.

$$\begin{aligned}
\boldsymbol{v}_{k,\mathsf{init}} &= (\cdots, 1, 0, 0 \;\|\; \quad 0, \quad 0, \mathbf{0}),\\
\boldsymbol{v}_{k,q} &= (\cdots, -\boxed{0}- \;\|\; s_{k,f}[q], 0, \mathbf{0}),\\
\widetilde{\boldsymbol{v}}_{k,q} &= (\cdots, -\boxed{0}- \;\|\; s_{k,f}[q], 0, \mathbf{0}),
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{u}_{k,\mathsf{init}} &= (\cdots, \boxed{\ell_{k,\mathsf{init}}}, 0, 0, \;\|\; \quad 0, \quad 0, \mathbf{0}),\\
\boldsymbol{u}_{k,t<T,\mathrm{i}} &= (\cdots, \quad -0- \quad \;\|\; \boldsymbol{s_x}[t, \mathrm{i}], \quad 0, \mathbf{0}),\\
\widetilde{\boldsymbol{u}}_{k,T+1,\mathrm{i}} &= (\cdots, \quad -0- \quad \;\|\; \boldsymbol{s_x}[T+1, \mathrm{i}], 0, \mathbf{0}\;)
\end{aligned}$$

where all the label values $\{\ell_{k,t,\mathrm{i},q}\}$ are simulated and the initial label values are computed as follows

$$\begin{aligned}
\ell_{1,\mathsf{init}} &\leftarrow \mathsf{RevSamp}(M_1, \boldsymbol{x}, \boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{d} + \beta_1, \{\ell_{k,t,\mathrm{i},q}\}),\\
\ell_{k,\mathsf{init}} &\leftarrow \mathsf{RevSamp}(M_k, \boldsymbol{x}, \beta_k, \{\ell_{k,t,\mathrm{i},q}\}), \quad \text{for all } k > 1
\end{aligned}$$

From this hybrid we can traverse in the reverse direction all the way to the very first hybrid while keeping the private attribute as $\boldsymbol{d}$. We also rearrange the elements using the security of IPFE so that the distribution of the ciphertext does not change with the occurrence of the secret key whether it comes before or after the ciphertext. This is important for the public key $\mathsf{UAWS}^{\mathsf{L}}$. The formal security is discussed in Theorem 3.

From single key to full-fledge $\mathsf{UAWS}^{\mathsf{L}}$ The next and final goal is to bootstrap the single key, single ciphertext secure $\mathsf{UAWS}^{\mathsf{L}}$ to a public key $\mathsf{UAWS}^{\mathsf{L}}$ scheme that supports releasing many secret keys and ciphertexts. Observe that our secret key $\mathsf{UAWS}^{\mathsf{L}}$ already supports multiple keys and single ciphertext. However, it fails to remain secure if two ciphertexts are published. This is because the piecewise security of AKGS can not be guaranteed if the label functions are reused. Our bootstrapping procedure takes inspiration from prior works [38, 62], that is to sample a random multiplier $s \leftarrow \mathbb{Z}_p$ at the time of encryption, which will randomize the label values in the exponent of $\mathbb{G}_2$. In particular, using IPFE security the random multiplier s is moved to the secret key vectors where the DDH assumption ensures that $s\boldsymbol{\ell}_{k,t,\mathfrak{i},q}$'s are pseudorandom in the exponent of $\mathbb{G}_2$. To upgrade the scheme into the public key setting, we employ the Slotted IPFE that enables encrypting into the public slots using the public key whereas the function hiding security still holds in the private slots. We describe below our public key $\mathsf{UAWS}^{\mathsf{L}}$ scheme.

$$\begin{aligned}
\mathsf{SK}_{\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}}} &= \{\mathsf{IPFE.SK}_{\boldsymbol{v}_{\mathsf{pad}}}\mathsf{IPFE.SK}_{\boldsymbol{v}_{k,\mathsf{init}}}, \mathsf{IPFE.SK}_{\boldsymbol{v}_{k,q}}, \widetilde{\mathsf{IPFE.SK}}_{\widetilde{\boldsymbol{v}}_{k,q}}\}_{k\in\mathcal{I}_{\boldsymbol{M}}} : \alpha \leftarrow \mathbb{Z}_p \\
[\![\boldsymbol{v}_{k,\mathsf{init}}]\!]_2 &= [\![(\quad 0, \quad \alpha, \quad 0, \quad 0, \quad 0, \quad \| \ \mathbf{0} \ )]\!]_2, \\
[\![\boldsymbol{v}_{k,\mathsf{init}}]\!]_2 &= [\![(\pi_k(1,k), \ 0, \ -\boldsymbol{r}_{k,f}[1], \quad \beta_k, \quad 0, \quad \| \ \mathbf{0} \ )]\!]_2, \\
[\![\boldsymbol{v}_{k,q}]\!]_2 &= [\![(\pi_k(1,k), \ 0, \ -\boldsymbol{r}_{k,f}[q], \quad 0, \quad (\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q] \ \| \ \mathbf{0} \ )]\!]_2, \\
[\![\widetilde{\boldsymbol{v}}_{k,q}]\!]_2 &= [\![(\pi_k(1,k), \ 0, \ -\boldsymbol{r}_{k,f}[q], \ \alpha\boldsymbol{y}_{k,\mathsf{acc}}[q] \quad \| \ \mathbf{0} \ )]\!]_2
\end{aligned}$$

$$\begin{aligned}
\mathsf{CT}_{\boldsymbol{x}} &= \{\mathsf{IPFE.CT}_{\boldsymbol{u}_{k,\mathsf{init}}}, \mathsf{IPFE.CT}_{\boldsymbol{u}_{k,t<T,\mathfrak{i}}}, \widetilde{\mathsf{IPFE.CT}}_{\widetilde{\boldsymbol{u}}_{k,T+1,\mathfrak{i}}}\}_k : s \leftarrow \mathbb{Z}_p \\
[\![\boldsymbol{u}_{\mathsf{pad}}]\!]_1 &= [\![(\quad 0, \quad s, \quad 0, \quad 0, \quad 0, \quad \| \ \perp \ )]\!]_1, \\
[\![\boldsymbol{u}_{k,\mathsf{init}}]\!]_1 &= [\![(\rho_k(-k,1), \ 0, \ s\cdot\boldsymbol{r}_{\boldsymbol{x}}[(0,1,1,\mathbf{0}_S)], \quad s, \quad 0, \quad \| \ \perp \ )]\!]_1, \\
[\![\boldsymbol{u}_{k,t<T,\mathfrak{i}}]\!]_1 &= [\![(\rho_k(-k,1), \ 0, \quad s\cdot\boldsymbol{r}_{\boldsymbol{x}}[t-1,\mathfrak{i}], \quad 0, \quad s\cdot c_\tau(\boldsymbol{x};\boldsymbol{r}_{\boldsymbol{x}}) \ \| \ \perp \ )]\!]_1, \\
[\![\widetilde{\boldsymbol{u}}_{k,T+1,\mathfrak{i}}]\!]_1 &= [\![(\rho_k(-k,1), \ 0, \quad s\cdot\boldsymbol{r}_{\boldsymbol{x}}[T,\mathfrak{i}], \quad s\cdot\boldsymbol{z}[k] \quad \| \ \perp \ )]\!]_1
\end{aligned}$$

The slots at the left/right of " $\|$ " are public/private. The ciphertexts are computed using only the public slots and the private slots are utilized only in the security analysis. At a very high level, we utilize the triple-slot encryption technique devised in [38] to simulate the pre-challenge secret keys with a dummy vector encoded into the ciphertext and hardwire the functional value into the post-challenge secret keys. As mentioned earlier, the triple-slot encryption technique [38] was devised for the non-uniform model which crucially uses the fact that the garbling randomness can be (fully) sampled in the key generation process. This does not hold in our setting. Thus, we design a more advanced three-slot encryption technique that is compatible with *distributed randomness* of AKGS garbling procedure. More specifically, we add one additional hidden subspace in order to realize such sophisticated mechanism for Logspace Turing machines. This additional subspace enables us to simulate the post-ciphertext secret keys with distributed randomness. However, shuttle technical challenges still remain to be overcome due to the structure of AKGS for Logspace Turing machines. We prove the security of the scheme in Theorem 4.

3 Preliminaries

In this section, we provide the necessary definitions and backgrounds that will be used in the sequence.

Notations We denote by λ the security parameter that belongs to the set of natural number $\mathbb{N}$ and 1^λ denotes its unary representation. We use the notation $s \leftarrow S$ to indicate the fact that s is sampled uniformly at random from the finite set S. For a distribution $\mathcal{X}$, we write $x \leftarrow \mathcal{X}$ to denote that x is sampled at random according to the distribution $\mathcal{X}$. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ is said to be a negligible function of λ, if for every $c \in \mathbb{N}$ there exists a $\lambda_c \in \mathbb{N}$ such that for all $\lambda > \lambda_c$, $|\mathsf{negl}(\lambda)| < \lambda^{-c}$.

Let Expt be an interactive security experiment played between a challenger and an adversary, which always outputs a single bit. We assume that $\mathsf{Expt}^{\mathsf{C}}_{\mathcal{A}}$ is a function of λ and it is parametrized by an adversary $\mathcal{A}$ and a cryptographic protocol C. Let $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{C},0}$ and $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{C},1}$ be two such experiment. The experiments are computationally/statistically indistinguishable if for any PPT/computationally unbounded adversary $\mathcal{A}$ there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{C}}(\lambda) = \left|\Pr\left[1 \leftarrow \mathsf{Expt}_{\mathcal{A}}^{\mathsf{C},0}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathsf{Expt}_{\mathcal{A}}^{\mathsf{C},1}(1^\lambda)\right]\right| < \mathsf{negl}(\lambda)$$

We write $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{C},0} \overset{c}{\approx} \mathsf{Expt}_{\mathcal{A}}^{\mathsf{C},1}$ if they are *computationally indistinguishable* (or simply *indistinguishable*). Similarly, $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{C},0} \overset{s}{\approx} \mathsf{Expt}_{\mathcal{A}}^{\mathsf{C},1}$ means *statistically indistinguishable* and $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{C},0} \equiv \mathsf{Expt}_{\mathcal{A}}^{\mathsf{C},1}$ means they are *identically* distributed.

Sets and indexing For $n \in \mathbb{N}$, we denote $[n]$ the set $\{1, 2, \ldots, n\}$ and for $n, m \in \mathbb{N}$ with $n < m$, we denote $[n, m]$ be the set $\{n, n+1, \ldots, m\}$. We use lowercase boldface, e.g., $\boldsymbol{v}$, to denote column vectors in $\mathbb{Z}_p^n$ and uppercase boldface, e.g., $\mathbf{M}$, to denote matrices in $\mathbb{Z}_p^{n\times m}$ for $p, n, m \in \mathbb{N}$. The i-th component of a vector $\boldsymbol{v} \in \mathbb{Z}_p^n$ is written as $\boldsymbol{v}[i]$ and the (i, j)-th element of a matrix $\mathbf{M} \in \mathbb{Z}_p^{n\times m}$ is denoted by $\mathbf{M}[i, j]$. The transpose of a matrix $\mathbf{M}$ is denoted by $\mathbf{M}^\top$ such that $\mathbf{M}^\top[i, j] = \mathbf{M}[j, i]$. To write a vector of length n with all zero elements, we write $\mathbf{0}_n$ or simply $\mathbf{0}$ when the length is clear from the context. Let $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{Z}_p^n$, then the inner product between the vectors is denoted as $\boldsymbol{u}\cdot\boldsymbol{v} = \boldsymbol{u}^\top\boldsymbol{v} = \sum_{i\in[n]} \boldsymbol{u}[i]\boldsymbol{v}[i] \in \mathbb{Z}_p$. We define *generalized* inner product between two vectors $\boldsymbol{u} \in \mathbb{Z}_p^{\mathcal{I}_1}$, $\boldsymbol{v} \in \mathbb{Z}_p^{\mathcal{I}_2}$ by $\boldsymbol{u}\cdot\boldsymbol{v} = \sum_{i\in\mathcal{I}_1\cap\mathcal{I}_2} \boldsymbol{u}[i]\boldsymbol{v}[i]$.

Tensor products Let $\boldsymbol{u} \in \mathbb{Z}_p^{\mathcal{I}_1}$ and $\boldsymbol{v} \in \mathbb{Z}_p^{\mathcal{I}_2}$ be two vectors, their Kronecker product $\boldsymbol{w} = \boldsymbol{u} \otimes \boldsymbol{v}$ is a vector in $\mathbb{Z}_p^{\mathcal{I}_1\times\mathcal{I}_2}$ with entries defined by $\boldsymbol{w}[(i, j)] = \boldsymbol{u}[i]\boldsymbol{v}[j]$. For two matrices $\mathbf{M}_1 \in \mathbb{Z}_p^{\mathcal{I}_1\times\mathcal{I}_2}$ and $\mathbf{M}_1 \in \mathbb{Z}_p^{\mathcal{I}'_1\times\mathcal{I}'_2}$,their Kronecker product $\mathbf{M} = \mathbf{M} = \mathbf{M}_1 \otimes \mathbf{M}_2$ is a matrix in $\mathbb{Z}_p^{(\mathcal{I}_1\times\mathcal{I}'_1)\times\mathcal{I}_2\times\mathcal{I}'_2}$ with entries defined by $\mathbf{M}[(i_1, i'_1), (i_2, i'_2)] = \mathbf{M}_1[i_1, i_2]\mathbf{M}_2[i'_1, i'_2]$.

Currying Currying is the product of partially applying a function or specifying part of the indices of a vector/matrices, which yields another function with fewer arguments or another vector/matrix with fewer indices. We use the usual syntax for evaluating a function or indexing into a vector/matrix, except that unspecified variables are represented by "$\llcorner\!\lrcorner$". For example, let $\mathbf{M} \in \mathbb{Z}_p^{([\mathcal{I}_1]\times[\mathcal{I}_2])\times([\mathcal{J}_1]\times[\mathcal{J}_2])}$ and $i_1 \in \mathcal{I}_1$, $j_2 \in \mathcal{J}_2$, then $\mathbf{M}[(i_1, \llcorner\!\lrcorner), (\llcorner\!\lrcorner, j_2)]$ is a matrix $\mathbf{N} \in \mathbb{Z}_p^{[\mathcal{I}_2]\times[\mathcal{J}_2]}$ such that $\mathbf{N}[i_2, j_1] = \mathbf{M}[(i_1, i_2), (j_1, j_2)]$ for all $i_2 \in \mathcal{I}_2$, $j_1 \in \mathcal{J}_1$.

Coefficient vector Let $f : \mathbb{Z}_p^{\mathcal{I}} \rightarrow \mathbb{Z}_p$ be an affine function with coefficient vector $\mathbf{f} \in \mathbb{Z}_p^{\mathcal{S}}$ for $\mathcal{S} = \{\mathsf{const}\} \cup \{\mathsf{coef}_i |\ i \in \mathcal{I}\}$. Then for any $\boldsymbol{x} \in \mathbb{Z}_p^{\mathcal{I}}$, we have $f(\boldsymbol{x}) = \mathbf{f}[\mathsf{const}] + \sum_{i\in\mathcal{I}} \mathbf{f}[\mathsf{coef}_i]\boldsymbol{x}[i]$.

3.1 Bilinear groups and hardness assumptions

We use a pairing group generator $\mathcal{G}$ that takes as input 1^λ and outputs a tuple $\mathsf{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathrm{T}, g_1, g_2, e)$ where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathrm{T}$ are groups of prime order $p = p(\lambda)$ and g_i is a generator of the group $\mathbb{G}_i$ for $i \in \{1, 2\}$. The map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_\mathrm{T}$ satisfies the following properties:

- *bilinear*: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all $a, b \in \mathbb{Z}_p$.
- *non-degenerate*: $e(g_1, g_2)$ generates $\mathbb{G}_\mathrm{T}$.

The group operations in $\mathbb{G}_i$ for $i \in \{1, 2, \mathrm{T}\}$ and the map e are efficiently computable in deterministic polynomial time in the security parameter λ. For a matrix $\mathbf{A}$ and each $i \in$

$\{1, 2, \mathrm{T}\}$, we use the notation $[\![\mathbf{A}]\!]_i$ to denote $g_i^{\mathbf{A}}$ where the exponentiation is element-wise. The group operation is written additively while using the bracket notation, i.e. $[\![\mathbf{A}+\mathbf{B}]\!]_i = [\![\mathbf{A}]\!]_i + [\![\mathbf{B}]\!]_i$ for matrices $\mathbf{A}$ and $\mathbf{B}$. Observe that, given $\mathbf{A}$ and $[\![\mathbf{B}]\!]_i$, we can efficiently compute $[\![\mathbf{AB}]\!]_i = \mathbf{A} \cdot [\![\mathbf{B}]\!]_i$. We write the pairing operation multiplicatively, i.e. $e([\![\mathbf{A}]\!]_1, [\![\mathbf{B}]\!]_2) = [\![\mathbf{A}]\!]_1 [\![\mathbf{B}]\!]_2 = [\![\mathbf{AB}]\!]_{\mathrm{T}}$.

Assumption 1 (Symmetric external Diffie–Hellman assumption) We say that the SXDH assumption holds in a pairing group $\mathsf{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_{\mathrm{T}}, g_1, g_2, e)$ of order p, if the DDH assumption holds in $\mathbb{G}_i$, i.e., $\{[\![a]\!]_i, [\![b]\!]_i, [\![ab]\!]_i\} \approx \{[\![a]\!]_i, [\![b]\!]_i, [\![c]\!]_i\}$ for $i \in \{1, 2, \mathrm{T}\}$ and $a, b, c \leftarrow \mathbb{Z}_p$.

3.2 Turing machine formulation

In this subsection, we describe the main computational model of this work, which is Turing machines with a read-only input and a read-write work tape. This type of Turing machines are used to handle decision problems belonging to space-bounded complexity classes such as Logspace predicates. We define below Turing machines with time complexity T and space complexity S. The Turing machine can either accept or reject an input string within this time/space bound. We also stick to the binary alphabet for the shake of simplicity.

Definition 1 (Turing machine with time/space bound computation) [62] A (deterministic) Turing machine over $\{0, 1\}$ is a tuple $M = (Q, \boldsymbol{y}_{\mathsf{acc}}, \delta)$, where $Q \geq 1$ is the number of states (we use $[Q]$ as the set of states and 1 as the initial state), $\boldsymbol{y}_{\mathsf{acc}} \in \{0, 1\}^Q$ indicates whether each state is accepting, and

$$\begin{aligned} \delta : [Q] \times \{0, 1\} \times \{0, 1\} &\to [Q] \times \{0, 1\} \times \{0, \pm 1\} \times \{0, \pm 1\}, \\ (q, x, w) &\mapsto (q', w', \Delta i, \Delta j) \end{aligned}$$

is the state transition function, which, given the current state q, the symbol x on the input tape under scan, and the symbol w on the work tape under scan, specifies the new state q', the symbol w' overwriting w, the direction Δi to which the input tape pointer moves, and the direction Δj to which the work tape pointer moves. The machine is required to hang (instead of halting) once it reaches on the accepting state, i.e., for all $q \in [Q]$ such that $\boldsymbol{y}_{\mathsf{acc}}[q] = 1$ and all $x, w \in \{0, 1\}$, it holds that $\delta(q, x, w) = (q, w, 0, 0)$.

For input length $N \geq 1$ and space complexity bound $S \geq 1$, the set of *internal configurations* of M is

$$\mathcal{C}_{M,N,S} = [N] \times [S] \times \{0, 1\}^S \times [Q],$$

where $(i, j, \boldsymbol{W}, q) \in \mathcal{C}_{M,N,S}$ specifies the input tape pointer $i \in [N]$, the work tape pointer $j \in [S]$, the content of the work tape $\boldsymbol{W} \in \{0, 1\}^S$ and the machine state $q \in [Q]$.

For any bit-string $\boldsymbol{x} \in \{0, 1\}^N$ for $N \geq 1$ and time/space complexity bounds $T, S \geq 1$, the machine M accepts $\boldsymbol{x}$ within time T and space S if there exists a sequence of internal configurations (*computation* path of T steps) $c_0, \ldots, c_T \in \mathcal{C}_{M,N,S}$ with $c_t = (i_t, j_t, \boldsymbol{W}_t, q_t)$ such that

$$i_0 = 1, j_0 = 1, \boldsymbol{W}_0 = \mathbf{0}_S, q_0 = 1 \text{(initial configuration)}$$

$$\text{for } 0 \leq t < T \begin{cases} \delta(q_t, \boldsymbol{x}[i_t], \boldsymbol{W}_t[j_t]) & = (q_{t+1}, \boldsymbol{W}_{t+1}[j_t], i_{t+1} - i_t, j_{t+1} - j_t), \\ \boldsymbol{W}_{t+1}[j] & = \boldsymbol{W}_t[j] \text{ for all } j \neq j_t \ \text{(valid transitions);} \end{cases}$$

$$\boldsymbol{y}_{\mathsf{acc}}[q_T] = 1 \ \text{(accepting).}$$

Denote by $M|_{N,T,S}$ the function $\{0,1\}^N \to \{0,1\}$ mapping $\boldsymbol{x}$ to whether M accepts $\boldsymbol{x}$ in time T and space S. Define $\mathsf{TM} = \{M \mid M$ is a Turing machine$\}$ to be the set of all Turing machines.

Note that, the above definition does not allow the Turing machines moving off the input/work tape. For instance, if δ specifies moving the input pointer to the left/right when it is already at the leftmost/rightmost position, there is no valid next internal configuration. This type of situation can be handled by encoding the input string described in [62]. The problem of moving off the work tape to the left can be managed similarly, however, moving off the work tape to the right is undetectable by the machine, and this is intended due to the space bound. That is, when the space bound is violated, the input is *silently* rejected.

3.3 Functional encryption for unbounded attribute-weighted sum for Turing machines

We formally present the syntax of FE for unbounded attribute-weighted sum (AWS) and define adaptive simulation security of the primitive. We consider the set of all Turing machines $\mathsf{TM} = \{M \mid M$ is a Turing machine$\}$ with time bound T and space bound S.

Definition 2 (The AWS functionality for Turing machines) For any $n, N \in \mathbb{N}$, the class of attribute-weighted sum functionalities is defined as

$$\left\{ \begin{array}{l|c} ((\boldsymbol{x} \in \{0,1\}^N, 1^T, 1^{2^S}), \boldsymbol{z} \in \mathbb{Z}_p^n) \mapsto \boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{z} \text{ where} & \\ & N, T, S \geq 1, \\ \boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{z} = \sum_{k \in \mathcal{I}_{\boldsymbol{M}}} \boldsymbol{z}[k] \cdot M_k(\boldsymbol{x}) & M_k \in \mathsf{TM}\ \forall k \in [n], \\ & \mathcal{I}_{\boldsymbol{M}} \subseteq [n] \text{ with } |\mathcal{I}_{\boldsymbol{M}}| \geq 1 \end{array} \right\}$$

Definition 3 (Functional encryption for attribute-weighted sum) An unbounded-slot FE for unbounded attribute-weighted sum associated to the set of Turing machines TM and the message space $\mathbb{M}$ consists of four PPT algorithms defined as follows:

$\mathsf{Setup}(1^\lambda)$ The setup algorithm takes as input a security parameter and outputs the master secret-key MSK and the master public-key MPK.

$\mathsf{KeyGen}(\mathsf{MSK}, (\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}}))$ The key generation algorithm takes as input MSK and a tuple of Turing machines $\boldsymbol{M} = (M_k)_{k \in \mathcal{I}_{\boldsymbol{M}}}$. It outputs a secret-key $\mathsf{SK}_{(\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}})}$ and makes $(\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}})$ available publicly.

$\mathsf{Enc}(\mathsf{MPK}, ((\boldsymbol{x}_i, 1^{T_i}, 1^{2^{S_i}}), \boldsymbol{z}_i)_{i \in [\mathcal{N}]})$ The encryption algorithm takes as input MPK and a message consisting of $\mathcal{N}$ number of public–private pair of attributes $(\boldsymbol{x}_i, \boldsymbol{z}_i) \in \mathbb{M}$ such that the public attribute $\boldsymbol{x}_i \in \{0,1\}^{N_i}$ for some $N_i \geq 1$ with time and space bounds given by $T_i, S_i \geq 1$, and the private attribute $\boldsymbol{z}_i \in \mathbb{Z}_p^{n_i}$. It outputs a ciphertext $\mathsf{CT}_{(\boldsymbol{x}_i, T_i, S_i)}$ and makes $(\boldsymbol{x}_i, T_i, S_i)_{i \in [\mathcal{N}]}$ available publicly.

$\mathsf{Dec}((\mathsf{SK}_{(\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}})}, (\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}})), (\mathsf{CT}_{(\boldsymbol{x}_i, T_i, S_i)}, (\boldsymbol{x}_i, T_i, S_i)_{i \in [\mathcal{N}]}))$ The decryption algorithm takes as input $\mathsf{SK}_{(\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}})}$ along with the tuple of Turing machines and index sets $(\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}})$, and a ciphertext $\mathsf{CT}_{(\boldsymbol{x}_i, T_i, S_i)}$ along with a collection of associated public attributes $(\boldsymbol{x}_i, T_i, S_i)_{i \in [\mathcal{N}]}$. It outputs a value in $\mathbb{Z}_p$ or $\perp$.

Correctness The unbounded-slot FE for unbounded attribute-weighted sum is said to be correct if for all $((\boldsymbol{x}_i \in \{0,1\}^{N_i}, 1^{T_i}, 1^{2^{S_i}}), \boldsymbol{z}_i \in \mathbb{Z}_p^{n_i})_{i \in [\mathcal{N}]}$ and for all $(\boldsymbol{M} = (M_k)_{k \in \mathcal{I}_{\boldsymbol{M}}}, \mathcal{I}_{\boldsymbol{M}})$, we get

$$\Pr\left[\begin{array}{c} \mathsf{Dec}((\mathsf{SK}_{(\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}})}, (\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}})), (\mathsf{CT}_{(\boldsymbol{x}_i, T_i, S_i)}, (\boldsymbol{x}_i, T_i, S_i)_{i \in [\mathcal{N}]})) = \sum_{i \in \mathcal{N}} \sum_{k \in \mathcal{I}_{\boldsymbol{M}}} M_k(\boldsymbol{x}_i) \boldsymbol{z}_i[k] : \\ (\mathsf{MSK}, \mathsf{MPK}) \leftarrow \mathsf{Setup}(1^\lambda), \mathsf{SK}_{(\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}})} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, (\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}})), \\ \mathsf{CT}_{(\boldsymbol{x}_i, T_i, S_i)} \leftarrow \mathsf{Enc}(\mathsf{MPK}, ((\boldsymbol{x}_i, 1^{T_i}, 1^{2^{S_i}}), \boldsymbol{z}_i)_{i \in [\mathcal{N}]}), \mathcal{I}_{\boldsymbol{M}} \subseteq [n_i]\ \forall i \in \mathcal{N} \end{array} \right] = 1$$

We now define the adaptively simulation-based security of FE for unbounded attribute-weighted sum for Turing machines.

Definition 4 (Adaptive simulation security) Let (Setup, KeyGen, Enc, Dec) be an unbounded-slot FE for unbounded attribute-weighted sum for TM and message space $\mathbb{M}$. The scheme is said to be $(\Phi_{\mathsf{pre}}, \Phi_{\mathsf{CT}}, \Phi_{\mathsf{post}})$-adaptively simulation secure if for any PPT adversary $\mathcal{A}$ making at most Φ_{CT} ciphertext queries and $\Phi_{\mathsf{pre}}, \Phi_{\mathsf{post}}$ secret key queries before and after the ciphertext queries respectively, we have $\mathsf{Expt}^{\mathsf{UAWS}}_{\mathcal{A},\mathsf{real}}(1^\lambda) \overset{c}{\approx} \mathsf{Expt}^{\mathsf{UAWS}}_{\mathcal{A},\mathsf{ideal}}(1^\lambda)$, where the experiments are defined as follows. Also, an unbounded-slot FE for attribute-weighted sums is said to be $(\mathsf{poly}, \Phi_{\mathsf{CT}}, \mathsf{poly})$-adaptively simulation secure if it is $(\Phi_{\mathsf{pre}}, \Phi_{\mathsf{CT}}, \Phi_{\mathsf{post}})$-adaptively simulation secure as well as Φ_{pre} and Φ_{post} are unbounded polynomials in the security parameter λ.

```
$\underline{\mathsf{Expt}^{\mathsf{UAWS}}_{\mathcal{A},\mathsf{real}}(1^\lambda)}$

1. $1^{\mathcal{N}} \leftarrow \mathcal{A}(1^\lambda)$;
2. $(\mathsf{MSK}, \mathsf{MPK}) \leftarrow \mathsf{Setup}(1^\lambda)$;
3. $(((\boldsymbol{x}_i, 1^{T_i}, 1^{S_i}), \boldsymbol{z}_i \in \mathbb{Z}_p^{n_i})_{i\in[\mathcal{N}]}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}(\mathsf{MSK},\cdot)}}(\mathsf{MPK})$;
4. $\mathsf{CT}_{(\boldsymbol{x}_i,T_i,S_i)} \leftarrow \mathsf{Enc}(\mathsf{MPK}, ((\boldsymbol{x}_i, 1^{T_i}, 1^{2^{S_i}}), \boldsymbol{z}_i)_{i\in[\mathcal{N}]})$;
5. return $\mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}(\mathsf{MSK},\cdot)}}(\mathsf{MPK}, \mathsf{CT})$

$\underline{\mathsf{Expt}^{\mathsf{UAWS}}_{\mathcal{A},\mathsf{ideal}}(1^\lambda)}$

1. $1^{N} \leftarrow \mathcal{A}(1^\lambda)$;
2. $(\mathsf{MSK}^*, \mathsf{MPK}) \leftarrow \mathsf{Setup}^*(1^\lambda, 1^N)$;
3. $(((\boldsymbol{x}_i, 1^{T_i}, 1^{S_i}), \boldsymbol{z}_i \in \mathbb{Z}_p^{n_i})_{i\in[\mathcal{N}]}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}_0^*(\mathsf{MSK}^*,\cdot)}}(\mathsf{MPK})$
4. $\mathsf{CT}_{(\boldsymbol{x}_i,T_i,S_i)} \leftarrow \mathsf{Enc}^*(\mathsf{MPK}, \mathsf{MSK}^*, (\boldsymbol{x}_i, 1^{T_i}, 1^{S_i}, n_i)_{i\in[\mathcal{N}]}, \mathcal{V})$;
5. return $\mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}_1^*(\mathsf{MSK}^*,(\boldsymbol{x}_i,1^{T_i},1^{S_i})_{i\in[\mathcal{N}]},\cdot,\cdot)}}(\mathsf{MPK}, \mathsf{CT}_{(\boldsymbol{x}_i,T_i,S_i)})$

$\underline{\mathcal{O}_{\mathsf{KeyGen}(\mathsf{MSK},\cdot)}}$

1. input: $(\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}})$
2. output: $\mathsf{SK}_{(\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}})}$

$\underline{\mathcal{O}_{\mathsf{KeyGen}_0^*(\mathsf{MSK}^*,\cdot)}}$

1. input: $(\boldsymbol{M}_\phi, \mathcal{I}_{\boldsymbol{M}_\phi})$ for $\phi \in [\Phi_{\mathsf{pre}}]$
2. output: $\mathsf{SK}_{(\boldsymbol{M}_\phi,\mathcal{I}_{\boldsymbol{M}_\phi})}$

$\underline{\mathsf{Enc}^*(\mathsf{MPK}, \mathsf{MSK}^*, (\boldsymbol{x}_i, 1^{T_i}, 1^{2^{S_i}}, n_i)_{i\in[\mathcal{N}]}, \cdot)}$

1. input: $\mathcal{V} = \{(\boldsymbol{M}_\phi, \mathcal{I}_{\boldsymbol{M}_\phi}), \sum_{i\in[\mathcal{N}]} \boldsymbol{M}_\phi(\boldsymbol{x}_i)^\top \boldsymbol{z}_i : \phi \in [\Phi_{\mathsf{pre}}]\}$
2. output: $\mathsf{CT}_{(\boldsymbol{x}_i,T_i,S_i)}$

$\underline{\mathcal{O}_{\mathsf{KeyGen}_1^*(\mathsf{MSK}^*,(\boldsymbol{x}_i^*)_{i\in[N]},\cdot,\cdot)}}$

1. input: $(\boldsymbol{M}_\phi, \mathcal{I}_{\boldsymbol{M}_\phi}), \sum_{i\in\mathcal{N}} \boldsymbol{M}_\phi(\boldsymbol{x}_i)^\top \boldsymbol{z}_i$ for $\phi \in [\Phi_{\mathsf{post}}]$
2. output: $\mathsf{SK}_{(\boldsymbol{M}_\phi,\mathcal{I}_{\boldsymbol{M}_\phi})}$
```

3.4 Function-hiding slotted inner product functional encryption

Definition 5 (Slotted inner product functional encryption) [62] Let $\mathsf{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathrm{T}, g_1, g_2, e)$ be a tuple of pairing groups of prime order p. A slotted inner product functional encryption (IPFE) scheme based on G consists of 5 efficient algorithms:

$\mathsf{IPFE.Setup}(1^\lambda, S_{\mathsf{pub}}, S_{\mathsf{priv}})$ The setup algorithm takes as input a security parameter λ and two disjoint index sets, the public slots S_{pub} and the private slots S_{priv}. It outputs the master secret-key IPFE.MSK and the master public-key IPFE.MPK. Let $S = S_{\mathsf{pub}} \cup S_{\mathsf{priv}}$ be the whole index set and $|S|, |S_{\mathsf{pub}}|, |S_{\mathsf{priv}}|$ denote the number of indices in S, S_{pub} and S_{priv} respectively.

$\mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}]\!]_2)$ The key generation algorithm takes as input IPFE.MSK and a vector $[\![\boldsymbol{v}]\!]_2 \in \mathbb{G}_2^{|S|}$. It outputs a secret-key IPFE.SK for $\boldsymbol{v} \in \mathbb{Z}_p^{|S|}$.

$\mathsf{IPFE.Enc}(\mathsf{IPFE.MSK}, [\![\boldsymbol{u}]\!]_1)$ The encryption algorithm takes as input IPFE.MSK and a vector $[\![\boldsymbol{u}]\!]_1 \in \mathbb{G}_1^{|S|}$. It outputs a ciphertext IPFE.CT for $\boldsymbol{u} \in \mathbb{Z}_p^{|S|}$.

$\mathsf{IPFE.Dec}(\mathsf{IPFE.SK}, \mathsf{IPFE.CT})$ The decryption algorithm takes as input a secret-key IPFE.SK and a ciphertext IPFE.CT. It outputs an element from $\mathbb{G}_\mathrm{T}$.

$\mathsf{IPFE.SlotEnc}(\mathsf{IPFE.MPK}, [\![\boldsymbol{u}]\!]_1)$ The slot encryption algorithm takes as input IPFE.MPK and a vector $[\![\boldsymbol{u}]\!]_1 \in \mathbb{G}_1^{|S_{\mathsf{pub}}|}$. It outputs a ciphertext IPFE.CT for $(\boldsymbol{u}||\boldsymbol{0}_{|S_{\mathsf{priv}}|}) \in \mathbb{Z}_p^{|S|}$.

Correctness The correctness of a slotted IPFE scheme requires the following two properties.

– Decryption Correctness: The slotted IPFE is said to satisfy decryption correctness if for all $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{Z}_p^{|S|}$, we have

$$\Pr\left[\begin{array}{l}\mathsf{Dec}(\mathsf{IPFE.SK}, \mathsf{IPFE.CT}) = [\![\boldsymbol{v} \cdot \boldsymbol{u}]\!]_{\mathrm{T}} : \\ (\mathsf{IPFE.MSK}, \mathsf{IPFE.MPK}) \leftarrow \mathsf{Setup}(1^\lambda, S_{\mathsf{pub}}, S_{\mathsf{priv}}), \\ \mathsf{IPFE.SK} \leftarrow \mathsf{KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}]\!]_2), \\ \mathsf{IPFE.CT} \leftarrow \mathsf{Enc}(\mathsf{IPFE.MSK}, [\![\boldsymbol{u}]\!]_1)\end{array}\right] = 1$$

– Slot-Mode Correctness: The slotted IPFE is said to satisfy the slot-mode correctness if for all vectors $\boldsymbol{u} \in \mathbb{Z}_p^{|S_{\mathsf{pub}}|}$, we have

$$\left\{\begin{array}{l}(\mathsf{IPFE.MSK}, \mathsf{IPFE.MPK}, \mathsf{IPFE.CT}) : \\ (\mathsf{IPFE.MSK}, \mathsf{IPFE.MPK}) \leftarrow \mathsf{Setup}(1^\lambda, S_{\mathsf{pub}}, S_{\mathsf{priv}}), \\ \mathsf{IPFE.CT} \leftarrow \mathsf{Enc}(\mathsf{IPFE.MSK}, [\![\boldsymbol{u} || \boldsymbol{0}_{|S_{\mathsf{priv}}|}]\!]_1)\end{array}\right\}$$

$$\equiv \left\{\begin{array}{l}(\mathsf{IPFE.MSK}, \mathsf{IPFE.MPK}, \mathsf{IPFE.CT}) : \\ (\mathsf{IPFE.MSK}, \mathsf{IPFE.MPK}) \leftarrow \mathsf{Setup}(1^\lambda, S_{\mathsf{pub}}, S_{\mathsf{priv}}), \\ \mathsf{IPFE.CT} \leftarrow \mathsf{SlotEnc}(\mathsf{IPFE.MPK}, [\![\boldsymbol{u}]\!]_1)\end{array}\right\}$$

Security Let (IPFE.Setup, IPFE.KeyGen, IPFE.Enc, IPFE.Dec, IPFE.SlotEnc) be a slotted IPFE. The scheme is said to be adaptively function-hiding secure if for all PPT adversary $\mathcal{A}$, we have $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{FH\text{-}IPFE}}(1^\lambda, 0) \overset{c}{\approx} \mathsf{Expt}_{\mathcal{A}}^{\mathsf{FH\text{-}IPFE}}(1^\lambda, 1)$, where the experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{FH\text{-}IPFE}}(1^\lambda, b)$ for $b \in \{0, 1\}$ is defined as follows:

```
$\underline{\mathsf{Expt}_{\mathcal{A}}^{\mathsf{FH\text{-}IPFE}}(1^\lambda, b)}$

1. $(S_{\mathsf{pub}}, S_{\mathsf{priv}}) \leftarrow \mathcal{A}(1^\lambda)$;
2. $(\mathsf{IPFE.MSK}, \mathsf{IPFE.MPK}) \leftarrow \mathsf{Setup}(1^\lambda, S_{\mathsf{pub}}, S_{\mathsf{priv}})$;
3. return $\mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}_b}(\cdot,\cdot), \mathcal{O}_{\mathsf{Enc}_b}(\cdot,\cdot)}(\mathsf{IPFE.MPK})$ if
     $\boldsymbol{v}_j^0|_{S_{\mathsf{pub}}} = \boldsymbol{v}_j^1|_{S_{\mathsf{pub}}}$ and $\boldsymbol{v}_j^0 \cdot \boldsymbol{u}_i^0 = \boldsymbol{v}_j^1 \cdot \boldsymbol{u}_i^1$
     for all $\{[\![\boldsymbol{v}_j^0]\!]_2, [\![\boldsymbol{v}_j^1]\!]_2\}_j$, $\{[\![\boldsymbol{u}_i^0]\!]_1, [\![\boldsymbol{u}_i^1]\!]_1\}_i$ queried
     by $\mathcal{A}$ to $\mathcal{O}_{\mathsf{KeyGen}_b}(\cdot,\cdot)$ and $\mathcal{O}_{\mathsf{Enc}_b}(\cdot,\cdot)$ respectively.

$\underline{\mathcal{O}_{\mathsf{KeyGen}_b}(\cdot,\cdot)}$:

1. input: $[\![\boldsymbol{v}_j^0]\!]_2, [\![\boldsymbol{v}_j^1]\!]_2 \in \mathbb{G}_2^{|S|}$
2. output
     $\mathsf{IPFE.SK}_j \leftarrow \mathsf{KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_j^b]\!]_2)$

$\underline{\mathcal{O}_{\mathsf{Enc}_b}(\cdot,\cdot)}$:

1. input: $[\![\boldsymbol{u}_i^0]\!]_1, [\![\boldsymbol{u}_i^1]\!]_1 \in \mathbb{G}_1^{|S|}$
2. output
     $\mathsf{IPFE.CT}_i \leftarrow \mathsf{Enc}(\mathsf{IPFE.MSK}, [\![\boldsymbol{u}_i^b]\!]_1)$
```

where $\boldsymbol{v}_j|_{S_{\mathsf{pub}}}$ represents the elements of $\boldsymbol{v}_j$ sitting at the indices in S_{pub}.

Lemma 1 [61, 62] *Let* $\mathsf{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_{\mathrm{T}}, g_1, g_2, e)$ *be a tuple of pairing groups of prime order* p *and* $k \geq 1$ *an integer constant. If* MDDH_k *holds in both groups* $\mathbb{G}_1, \mathbb{G}_2$*, then there is an adaptively function-hiding secure* IPFE *scheme based on* G.

3.5 Arithmetic key garbling scheme for Turing machines

Lin and Luo [62] introduced arithmetic key garbling scheme (AKGS). The notion of AKGS is an information theoretic primitive, inspired by randomized encodings [18] and partial garbling schemes [51]. It garbles a function $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$ (possibly of size $(m + 1)$) along with two secrets $z, \beta \in \mathbb{Z}_p$ and produces affine label functions $L_1, \ldots, L_{m+1} : \mathbb{Z}_p^n \to \mathbb{Z}_p$. Given f, an input $\boldsymbol{x} \in \mathbb{Z}_p^n$ and the values $L_1(\boldsymbol{x}), \ldots, L_{m+1}(\boldsymbol{x})$, there is an efficient algorithm which computes $zf(\boldsymbol{x}) + \beta$ without revealing any information about z and β. Lin and Luo [62] additionally design AKGS for Turing machines with time/space bounds. Many parts of this section are taken from the Sections 5 and 7.1 of [62]. Thus, the reader familiar with the notion of AKGS for Turing machines can skip this section. We define AKGS for the function class

$$\mathcal{F} = \{M|_{N,T,S} : \mathbb{Z}_p^N \to \mathbb{Z}_p, N, T, S \geq 1, p \text{ prime}\}$$

for the set of all time/space bounded Turing machine computations. We refer to [62] for a detailed discussion on the computation of Turing machines as a sequence of matrix multiplications, and the construction of AKGS for matrix multiplication.

Definition 6 (Arithmetic key garbling scheme (AKGS)) [62] An arithmetic garbling scheme (AKGS) for the function class $\mathcal{F}$, consists of two efficient algorithms:

$\mathsf{Garble}((M, 1^N, 1^T, 1^{2^S}, p), z, \beta)$ The garbling is a randomized algorithm that takes as input a tuple of a function $M|_{N,T,S}$ over $\mathbb{Z}_p$ from $\mathcal{F}$, an input length N, a time bound T, a space bound S with $N, T, S \geq 1$, a prime p, and two secret integers $z, \beta \in \mathbb{Z}_p$. It outputs a set of affine functions $L_{\mathsf{init}}, (L_{t,\theta})_{t\in[T+1],\theta\in\mathcal{C}_{M,N,S}} : \mathbb{Z}_p^N \to \mathbb{Z}_p$ which are called label functions that specifies how an input of length N is encoded as labels. Pragmatically, it outputs the coefficient vectors $\boldsymbol{\ell}_{\mathsf{init}}, (\boldsymbol{\ell}_{t,\theta})_{t\in[T+1],\theta\in\mathcal{C}_{M,N,S}}$.

$\mathsf{Eval}((M, 1^N, 1^T, 1^{2^S}, p), \boldsymbol{x}, \ell_{\mathsf{init}}, (\ell_{t,\theta})_{t\in[T+1],\theta\in\mathcal{C}_{M,N,S}})$ The evaluation is a deterministic algorithm that takes as input a function $M|_{N,T,S}$ over $\mathbb{Z}_p$ from $\mathcal{F}$, an input vector $\boldsymbol{x} \in \mathbb{Z}_p^N$ and the integers $\ell_{\mathsf{init}}, (\ell_{t,\theta})_{t\in[T+1],\theta\in\mathcal{C}_{M,N,S}} \in \mathbb{Z}_p$ which are supposed to be the values of the label functions at $\boldsymbol{x} \in \mathbb{Z}_p^N$. It outputs a value in $\mathbb{Z}_p$.

Correctness The AKGS is said to be correct if for all tuple $(M, 1^N, 1^T, 1^{2^S}, p)$, integers $z, \beta \in \mathbb{Z}_p$ and $\boldsymbol{x} \in \mathbb{Z}_p^N$, we have

$$\Pr\left[\begin{array}{l} \mathsf{Eval}((M, 1^N, 1^T, 1^{2^S}, p), \boldsymbol{x}, \ell_{\mathsf{init}}, (\ell_{t,\theta})_{t\in[T+1],\theta\in\mathcal{C}_{M,N,S}}) \\ = zM|_{N,T,S}(\boldsymbol{x}) + \beta : \\ (\boldsymbol{\ell}_{\mathsf{init}}, (\boldsymbol{\ell}_{t,\theta})_{t\in[T+1],\theta\in\mathcal{C}_{M,N,S}}) \leftarrow \mathsf{Garble}((M, 1^N, 1^T, 1^{2^S}, p), z, \beta), \\ \text{where } \ell \leftarrow L(\boldsymbol{x}) \end{array}\right] = 1$$

The scheme have *deterministic shape*, meaning that the number of label functions, $m = 1 + (T+1)NS2^SQ$, is determined solely by the tuple $(M, 1^N, 1^T, 1^{2^S}, p)$, independent of z, β and the randomness in Garble. The number of label functions m is called the *garbling size* of $M|_{N,T,S}$ under this scheme. For the sake of simpler representation, let us number the label values (or functions) as $1, \ldots, m$ in the lexicographical order where the first two label values are $\ell_{\mathsf{init}}, \ell_{(1,1,1,\mathbf{0}_S,1)}$ and the last label value is $\ell_{(T+1,N,S,\mathbf{1}^S,Q)}$.

Linearity The AKGS is said to be *linear* if the following conditions hold:

- $\mathsf{Garble}((M, 1^N, 1^T, 1^{2^S}, p), z, \beta)$ uses a uniformly random vector $\boldsymbol{r} \leftarrow \mathbb{Z}_p^m$ as its randomness, where m is determined solely by $(M, 1^N, 1^T, 1^{2^S}, p)$, independent of z, β.
- The coefficient vectors $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_m$ produced by $\mathsf{Garble}((M, 1^N, 1^T, 1^{2^S}, p), z, \beta)$ are linear in $(z, \beta, \boldsymbol{r})$.
- $\mathsf{Eval}((M, 1^N, 1^T, 1^{2^S}, p), \boldsymbol{x}, \ell_1, \ldots, \ell_m)$ is linear in $\ell_1, \ldots, \ell_m$.

For our UAWS, we consider the piecewise security notion of AKGS defined by Lin and Luo [62][1].

[1] The usual simulation-based security considered in previous works [38, 51] follows from the piecewise security of AKGS.

Definition 7 (Piecewise security of AKGS) [62] An $\mathsf{AKGS} = (\mathsf{Garble}, \mathsf{Eval})$ for the function class $\mathcal{F}$ is *piecewise* secure if the following conditions hold:

– The first label value is *reversely sampleable* from the other labels together with $(M, 1^N, 1^T, 1^{2^S}, p)$ and $\boldsymbol{x}$. This reconstruction is perfect even given all the other label functions. Formally, there exists an efficient algorithm $\mathsf{RevSamp}$ such that for all $M|_{N,T,S} \in \mathcal{F}, z, \beta \in \mathbb{Z}_p$ and $\boldsymbol{x} \in \mathbb{Z}_p^N$, the following distributions are identical:

$$\left\{(\ell_1, \boldsymbol{\ell}_2, \ldots, \boldsymbol{\ell}_m) : \begin{array}{l}(\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_m) \leftarrow \mathsf{Garble}((M, 1^N, 1^T, 1^{2^S}, p), z, \beta), \\ \ell_1 \leftarrow L_1(\boldsymbol{x})\end{array}\right\},$$

$$\left\{(\ell_1, \boldsymbol{\ell}_2, \ldots, \boldsymbol{\ell}_m) : \begin{array}{l}(\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_m) \leftarrow \mathsf{Garble}((M, 1^N, 1^T, 1^{2^S}, p), z, \beta), \\ \ell_j \leftarrow L_j(\boldsymbol{x}) \text{ for } j \in [2, m], \\ \ell_1 \leftarrow \mathsf{RevSamp}((M, 1^N, 1^T, 1^{2^S}, p), \boldsymbol{x}, zM|_{N,T,S}(\boldsymbol{x}) + \\ \hfill \beta, \ell_2, \ldots, \ell_m)\end{array}\right\}$$

– For the other labels, each is *marginally random* even given all the label functions after it. Formally, this means for all $M|_{N,T,S} \in \mathcal{F}, z, \beta \in \mathbb{Z}_p, \boldsymbol{x} \in \mathbb{Z}_p^n$ and all $j \in [2, m]$, the following distributions are identical:

$$\left\{(\ell_j, \boldsymbol{\ell}_{j+1}, \ldots, \boldsymbol{\ell}_m) : \begin{array}{l}(\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_m) \leftarrow \mathsf{Garble}((M, 1^N, 1^T, 1^{2^S}, p), z, \beta), \\ \ell_j \leftarrow L_j(\boldsymbol{x})\end{array}\right\},$$

$$\left\{(\ell_j, \boldsymbol{\ell}_{j+1}, \ldots, \boldsymbol{\ell}_m) : \begin{array}{l}(\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_m) \leftarrow \mathsf{Garble}((M, 1^N, 1^T, 1^{2^S}, p), z, \beta), \\ \ell_j \leftarrow \mathbb{Z}_p\end{array}\right\}$$

We now define special structural properties of AKGS as given in [62], related to the piecewise security of it.

Definition 8 (Special piecewise security of AKGS, [62]) An $\mathsf{AKGS} = (\mathsf{Garble}, \mathsf{Eval})$ for a function class $\mathcal{F}$ is *special* piecewise secure if for any $(M, 1^N, 1^T, 1^{2^S}, p) \in \mathcal{F}, z, \beta \in \mathbb{Z}_p$ and $\boldsymbol{x} \in \mathbb{Z}_p^N$, it has the following special form:

– The first label value ℓ_1 is always non-zero, i.e., $\mathsf{Eval}((M, 1^N, 1^T, 1^{2^S}, p), \boldsymbol{x}, 1, 0, \ldots, 0) \neq 0$ where we take $\ell_1 = 1$ and $\ell_j = 0$ for $1 < j \leq m$.

– Let $\boldsymbol{r} \leftarrow \mathbb{Z}_p^m$ be the randomness used in $\mathsf{Garble}((M, 1^N, 1^T, 1^{2^S}, p), z, \beta)$. For all $j \in [2, m]$, the label function L_j produced by Garble $((M, 1^N, 1^T, 1^S, p), z, \beta; \boldsymbol{r})$ can be written as

$$L_j(\boldsymbol{x}) = k_j \boldsymbol{r}[j-1] + L'_j(\boldsymbol{x}; z, \beta, \boldsymbol{r}[j], \boldsymbol{r}[j+1], \ldots, \boldsymbol{r}[m])$$

where $k_j \in \mathbb{Z}_p$ is a non-zero constant (not depending on $\boldsymbol{x}, z, \beta, \boldsymbol{r}$) and L'_j is an affine function of $\boldsymbol{x}$ whose coefficient vector is linear in $(z, \beta, \boldsymbol{r}[j], \boldsymbol{r}[j+1], \ldots, \boldsymbol{r}[m])$. The component $\boldsymbol{r}[j-1]$ is called the randomizer of L_j and ℓ_j.

Lemma 2 [62] *A special piecewise secure* $\mathsf{AKGS} = (\mathsf{Garble}, \mathsf{Eval})$ *for a function class* $\mathcal{F}$ *is also piecewise secure. The* $\mathsf{RevSamp}$ *algorithm* (*required in piecewise security*) *obtained for a special piecewise secure* AKGS *is linear in* $\gamma, \ell_2, \ldots, \ell_{m+1}$ *and perfectly recovers* ℓ_1

even if the randomness of Garble *is not uniformly sampled. More specifically, we have the following:*

$$
\begin{aligned}
&\mathsf{Eval}((M, 1^N, 1^T, 1^{2^S}, p), \boldsymbol{x}, \ell_1, \ldots, \ell_m)\\
&= \ell_1 \mathsf{Eval}((M, 1^N, 1^T, 1^{2^S}, p), \boldsymbol{x}, 1, 0, \ldots, 0) + \mathsf{Eval}((M, 1^N, 1^T, 1^{2^S}, p), \boldsymbol{x}, 0, \ell_2, \ldots, \ell_m)\\
&\mathsf{RevSamp}((M, 1^N, 1^T, 1^{2^S}, p), \boldsymbol{x}, \gamma, \ell_2, \ldots, \ell_m)\\
&= (\mathsf{Eval}((M, 1^N, 1^T, 1^{2^S}, p), \boldsymbol{x}, 1, 0, \ldots, 0))^{-1}(\gamma - \mathsf{Eval}((M, 1^N, 1^T, 1^{2^S}, p), \boldsymbol{x}, 0, \ell_2, \ldots, \ell_m))
\end{aligned}
$$

Note that, Eq. (2) follows from the linearity of Eval and Eq. (2) ensures that RevSamp perfectly computes ℓ_1 (which can be verified by Eq. (2) with $\gamma = zM|_{N,T,S}(\boldsymbol{x}) + \beta$).

Lemma 3 [62] *A piecewise secure* AKGS = (Garble, Eval) *is also special piecewise secure after an appropriate change of variable for the randomness used by* Garble.

4 Construction of AKGS for the class $\mathcal{F}$

We now describe the AKGS construction for the function class $\mathcal{F}$ given by Lin and Luo [62]. Before going to the actual construction, we first represent the computation of Turing machines as a sequence of matrix multiplications.

Transition matrix Given a Turing machine $M = (Q, \boldsymbol{y}_{\mathsf{acc}}, \delta)$, upper bounds of time and space $T, S \geq 1$ and an input $\boldsymbol{x} \in \{0,1\}^N$ for some $N \geq 1$, we consider the length-Tcomputation path of M with input $\boldsymbol{x}$ and space bound S. Recall that the set of internal configuration is $\mathcal{C}_{M,N,S} = [N] \times [S] \times \{0,1\}^S \times [Q]$. An internal configuration $\theta = (i, j, \boldsymbol{W}, q) \in \mathcal{C}_{M,N,S}$ specifies that the input and work tape pointers are at position i and j respectively, the work tape has content $\boldsymbol{W}$, an the current state is q. In particular, the initial configuration is $(1, 1, \mathbf{0}_S, 1)$: the input/work tape pointers point to the first cell, the work tape is all-0, and the state is the initial state 1. An accepting configuration satisfies that $\boldsymbol{y}_{\mathsf{acc}}[q] = 1$.

We construct a transition matrix $\mathbf{M}_{N,S}(\boldsymbol{x}) \in \{0,1\}^{\mathcal{C}_{M,N,S} \times \mathcal{C}_{M,N,S}}$ such that $\mathbf{M}_{N,S}(\boldsymbol{x})[\theta, \theta'] = 1$ if and only if the internal configuration of M is θ' after 1 step of computation starting from internal configuration θ. According to how the Turing machine operates in each step depending on the transition function δ, the entries of $\mathbf{M}_{N,S}(\boldsymbol{x})$ are defined as follows:

$$
\begin{aligned}
&\mathbf{M}_{N,S}(\boldsymbol{x})[(i, j, \boldsymbol{W}, q), (i', j', \boldsymbol{W}', q')]\\
&= \begin{cases} 1, & \text{if } \delta(q, \boldsymbol{x}[i], \boldsymbol{W}[j]) = (q', \boldsymbol{W}'[j], i' - i, j' - j)\\ & \quad \text{and } \boldsymbol{W}'[j''] = \boldsymbol{W}[j''] \text{ for all } j'' \neq j;\\ 0, & \text{otherwise;} \end{cases}\\
&= \boldsymbol{x}[i] \times \begin{cases} 1, & \text{if } \delta(q, 1, \boldsymbol{W}[j]) = (q', \boldsymbol{W}'[j], i' - i, j' - j)\\ & \quad \text{and } \boldsymbol{W}'[j''] = \boldsymbol{W}[j''] \text{ for all } j'' \neq j;\\ 0, & \text{otherwise;} \end{cases}\\
&\quad + (1 - \boldsymbol{x}[i]) \times \begin{cases} 1, & \text{if } \delta(q, 0, \boldsymbol{W}[j]) = (q', \boldsymbol{W}'[j], i' - i, j' - j)\\ & \quad \text{and } \boldsymbol{W}'[j''] = \boldsymbol{W}[j''] \text{ for all } j'' \neq j;\\ 0, & \text{otherwise;} \end{cases}
\end{aligned}
$$

With the transition matrix, we can now write the computation of Turing machines as a sequence of matrix multiplication. We represent initial configurations using one-hot encoding—the internal configuration θ is represented by the basis vector $\boldsymbol{e}_\theta \in \{0,1\}^{\mathcal{C}_{M,N,S}}$ whose θ-entry is 1 and the other entries are 0. Observe that multiplying $\boldsymbol{e}_\theta^\top$ on the right by the transition matrix $\mathbf{M}_{N,S}(\boldsymbol{x})$ produces exactly the next internal configuration: if there is no valid internal configuration of M after 1 step of computation starting from θ, we have $\boldsymbol{e}_\theta^\top \mathbf{M}_{N,S}(\boldsymbol{x}) = \mathbf{0}$; otherwise, the next internal configuration θ' is unique and $\boldsymbol{e}_\theta^\top \mathbf{M}_{N,S}(\boldsymbol{x}) = \boldsymbol{e}_{\theta'}^\top$. The function $M|_{N,T,S}(\boldsymbol{x})$ can be written as

$$M|_{N,T,S}(\boldsymbol{x}) = \boldsymbol{e}_{(1,1,\mathbf{0}_S,1)}^\top (\mathbf{M}_{N,S}(\boldsymbol{x}))^T (\mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \boldsymbol{y}_{\mathsf{acc}})$$

where $\boldsymbol{e}_{(1,1,\mathbf{0}_S,1)}$ represents the initial internal configuration. The sequence of multiplication advances the computation by T steps and test whether the final internal configuration is an accepting state. We elaborate on the last step: The tensor product $\mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \boldsymbol{y}_{\mathsf{acc}}$ is a vector in $\{0,1\}^{\mathcal{C}_{M,N,S}}$ such that its $(i,j,\boldsymbol{W},q)$-the entry is 1 if and only if $\boldsymbol{y}_{\mathsf{acc}}[q] = 1$, i.e., q is an accepting state. Therefore, taking the inner product of $\boldsymbol{e}_{(1,1,\mathbf{0}_S,1)}^\top (\mathbf{M}_{N,S}(\boldsymbol{x}))^T = \boldsymbol{e}_{\theta'}^\top$ (θ' is the final internal configuration) or 0 with the tensor product indicates whether M accepts $\boldsymbol{x}$ within time T and space S.

Transition blocks We observe that the transition matrix has the following two useful properties:

– $\mathbf{M}_{N,S}(\boldsymbol{x})$ is affine in $\boldsymbol{x}$ when regarded as an integer matrix.
– $\mathbf{M}_{N,S}(\boldsymbol{x})$ has the following block structure. There is a finite set $\{\mathbf{M}_\tau\}_\tau$ of $Q \times Q$ matrices defined by the transition function δ, called *transition blocks*, such that for every $(i,j,\boldsymbol{W},q)$ and $(i',j',\boldsymbol{W}',q')$ in $[N]\times[S]\times\{0,1\}^S\times Q$, the submatrix $\mathbf{M}_{N,S}(\boldsymbol{x})[(i,j,\boldsymbol{W},\llcorner\!\lrcorner),(i',j',\boldsymbol{W}',\llcorner\!\lrcorner)]$ is either some $\mathbf{M}_\tau$ or $\mathbf{0}$.

Below we define the transition blocks.

Definition 9 Let $M = (Q, \boldsymbol{y}_{\mathsf{acc}}, \delta)$ be a Turing machine and $\mathcal{T} = \{0,1\}^3 \times \{0,\pm1\}^2$ the set of transition types. The transition blocks of M consists of 72 transition matrices $\mathbf{M}_\tau \in \{0,1\}^{Q\times Q}$ for $\tau = (x, w, w', \Delta i, \Delta j) \in \mathcal{T}$, each encoding the possible transitions among the states given the following information: the input tape symbol x under scan, the work tape symbol w under scan, the symbol w' overwriting w, the direction Di to which the input tape pointer moves, and the direction Dj to which the work tape pointer moves. Formally,

$$\mathbf{M}_{x,w,',\Delta i,\Delta j}[q,q'] = \begin{cases} 1, & \text{if } \delta(q,x,w) = (q',w',\Delta i,\Delta j); \\ 0, & \text{otherwise} \end{cases}$$

In $\mathbf{M}_{N,S}(\boldsymbol{x})$, each $Q \times Q$ block is either one of the transition blocks or $\mathbf{0}$:

$$\begin{aligned} &\mathbf{M}_{N,S}(\boldsymbol{x})[(i,j,\boldsymbol{W},\llcorner\!\lrcorner),(i',j',\boldsymbol{W}',\llcorner\!\lrcorner)] \\ &= \begin{cases} \mathbf{M}_{\boldsymbol{x}[i],\boldsymbol{W}[j],\boldsymbol{W}'[j],i'-i,j'-j}, & \text{if } i'-i,\ j'-j \in \{0,\pm1\} \text{ and} \\ & \boldsymbol{W}[j''] = \boldsymbol{W}'[j''] \text{ for all } j'' \neq j; \\ \mathbf{0}, & \text{otherwise} \end{cases} \end{aligned}$$

Observe further that in $\mathbf{M}_{N,S}(\boldsymbol{x})[(i,j,\boldsymbol{W},\llcorner\!\lrcorner),(\llcorner\!\lrcorner,\llcorner\!\lrcorner,\llcorner\!\lrcorner,\llcorner\!\lrcorner)]$, each transition block appears at most once.

AKGS for Turing machines. Above, we have represented the Turing machine computation as a sequence of matrix multiplication *over the integers*:

$$\begin{aligned} &M|_{N,T,S}(\boldsymbol{x}) \\ &= \boldsymbol{e}_{(1,1,\mathbf{0}_S,1)}^\top (\mathbf{M}_{N,S}(\boldsymbol{x}))^T \left(\mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \boldsymbol{y}_{\mathsf{acc}}\right) \text{ for } \boldsymbol{x} \in \{0,1\}^N \end{aligned}$$

We can formally extend $M|_{N,T,S} : \{0,1\}^N \to \{0,1\}$ to a $\mathbb{Z}_p^N \to \mathbb{Z}_p$ function using the same matrix multiplication formula, preserving its behavior when the input comes from $\{0,1\}^N$. When p is clear from the context, we use $M|_{N,T,S}$ to represent its extension over $\mathbb{Z}_p$. We now describe the construction of AKGS [62] for the Turing machine computations.

We consider the function class

$$\mathcal{F} = \left\{ M|_{N,T,S} : \mathbb{Z}_p^N \to \mathbb{Z}_p, N, T, S \geq 1, p \text{ prime} \right\}$$

which is the set of time/space bounded Turing machine computations. The AKGS $=$ (Garble, Eval) for the function class works as follows:

Garble$((M, 1^N, 1^T, 1^{2^S}, p), z, \beta)$ It takes a function $M|_{N,T,S}$ over $\mathbb{Z}_p$ from $\mathcal{F}$ and two secrets $z, \beta \in \mathbb{Z}_p$ as input. Suppose $M = (Q, \boldsymbol{y}_{\mathsf{acc}}, \delta)$, the algorithm samples $\boldsymbol{r}$ as the randomness by

$$\begin{aligned} &\text{for } t \in [0,T] : \boldsymbol{r}_t \leftarrow \mathbb{Z}_p^{\mathcal{C}_{M,N,S}} && \left(\mathcal{C}_{M,N,S} = [N] \times [S] \times \{0,1\}^S \times [Q]\right), \\ &\boldsymbol{r} \leftarrow \mathbb{Z}_p^{[0,T] \times \mathcal{C}_{M,N,S}}, && \boldsymbol{r}[t,i,j,\mathbf{W},q] = \boldsymbol{r}_t[(i,j,\boldsymbol{W},q)]. \end{aligned}$$

It computes the transition matrix $\mathbf{M}_{N,S}(\boldsymbol{x})$ as a function of $\boldsymbol{x}$ and defines the label functions by

$$\begin{aligned} L_{\mathsf{init}}(\boldsymbol{x}) &= \beta + \boldsymbol{e}_{(1,1,\mathbf{0}_S,1)}^{\top} \boldsymbol{r}_0, \\ \text{for } t \in [T] : \ (L_{t,\theta})_{\theta \in \mathcal{C}_{M,N,S}}(\boldsymbol{x}) &= -\boldsymbol{r}_{t-1} + \mathbf{M}_{N,S}(\boldsymbol{x}) \boldsymbol{r}_t, \\ (L_{T+1,\theta})_{\theta \in \mathcal{C}_{M,N,S}} &= -\boldsymbol{r}_T + z \mathbf{1}_{[N] \times [S] \times \{0,1\}^S} \otimes \boldsymbol{y}_{\mathsf{acc}}. \end{aligned}$$

It collects the coefficients of these label functions and returns them as $(\boldsymbol{\ell}_{\mathsf{init}}, (\boldsymbol{\ell}_{t,\theta})_{t \in [T+1], \theta \in \mathcal{C}_{M,N,S}})$.

Note: We show that Garble satisfies the required properties of a linear AKGS:

- The label functions are affine in $\boldsymbol{x}$: L_{init} and $L_{T+1,\theta}$ for all $\theta \in \mathcal{C}_{M,N,S}$ are constant with respect to $\boldsymbol{x}$. The rest are $L_{t,\theta}(\boldsymbol{x}) = (-\boldsymbol{r}_{t-1} + \mathbf{M}_{N,S}(\boldsymbol{x}) \boldsymbol{r}_t)[\theta]$. Since $\mathbf{M}_{N,S}(\boldsymbol{x})$ is affine in $\boldsymbol{x}$ and $\boldsymbol{r}_{t-1}, \boldsymbol{r}_t$ are constant with respect to $\boldsymbol{x}$, these label functions are also affine in $\boldsymbol{x}$.
- Shape determinism holds: The garbling size of $M|_{N,T,S}$ is $1 + (T+1)NS2^S Q$.
- Garble is linear in $z, \beta, \boldsymbol{x}$: The coefficients of the label functions are linear in $(z, \beta, \boldsymbol{x})$. Observe that $\mathbf{M}_{N,S}(\boldsymbol{x})$, $\boldsymbol{e}_{(1,1,\mathbf{0}_S,1)}$ and $\boldsymbol{y}_{\mathsf{acc}}$ are constant with respect to $(z, \beta, \boldsymbol{r})$, and z, β and $\boldsymbol{r}_t$ for all $t \in [0,T]$ are linear in $(z, \beta, \boldsymbol{x})$. By the definition of the label functions, their coefficients are linear in $(z, \beta, \boldsymbol{x})$.

Eval$((M, 1^N, 1^T, 1^{2^S}, p), \boldsymbol{x}, \ell_1, \ldots, \ell_m)$ It takes a function $M|_{N,T,S}$ over $\mathbb{Z}_p$ from $\mathcal{F}$, an input string $\boldsymbol{x} \in \mathbb{Z}_p^N$ and the labels as input. It first computes the transition matrix $\mathbf{M}_{N,S}(\boldsymbol{x})$ with $\boldsymbol{x}$ substituted into it and sets $\boldsymbol{\ell}_t = (\ell_{t,\theta})_{\theta \in \mathcal{C}_{M,N,S}}$ for $t \in [T+1]$. The algorithm computes and returns

$$\ell_{\mathsf{init}} + \boldsymbol{e}_{(1,1,\mathbf{0}_S,1)}^{\top} \sum_{t=1}^{T+1} (\mathbf{M}_{N,S}(\boldsymbol{x}))^{t-1} \boldsymbol{\ell}_t$$

Correctness Plugging $\ell_{t,\theta} = L_{t,\theta}(\boldsymbol{x})$ and the formula for $M|_{N,T,S}$ into the simulation, we find that it is a telescoping sum:

$$\begin{aligned}
&\boldsymbol{e}^{\top}_{(1,1,\mathbf{0}_S,1)} \sum_{t=1}^{T+1} (\mathbf{M}_{N,S}(\boldsymbol{x}))^{t-1} \boldsymbol{\ell}_t \\
&= \boldsymbol{e}^{\top}_{(1,1,\mathbf{0}_S,1)} \sum_{t=1}^{T+1} (\mathbf{M}_{N,S}(\boldsymbol{x}))^{t-1} (-\boldsymbol{r}_{t-1} + \mathbf{M}_{N,S}(\boldsymbol{x})\boldsymbol{r}_t) \\
&\quad + \boldsymbol{e}^{\top}_{(1,1,\mathbf{0}_S,1)} (\mathbf{M}_{N,S}(\boldsymbol{x}))^{T} (-\boldsymbol{r}_T + z\mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \boldsymbol{y}_{\mathsf{acc}}) \\
&= \boldsymbol{e}^{\top}_{(1,1,\mathbf{0}_S,1)} \sum_{t=1}^{T} (-(\mathbf{M}_{N,S}(\boldsymbol{x}))^{t-1} \boldsymbol{r}_{t-1} + (\mathbf{M}_{N,S}(\boldsymbol{x}))^{t} \boldsymbol{r}_t) \\
&\quad - \boldsymbol{e}^{\top}_{(1,1,\mathbf{0}_S,1)} (\mathbf{M}_{N,S}(\boldsymbol{x}))^{T} \boldsymbol{r}_T + zM|_{N,T,S}(\boldsymbol{x}) \\
&= -\boldsymbol{e}^{\top}_{(1,1,\mathbf{0}_S,1)} \boldsymbol{r}_0 + zM|_{N,T,S}(\boldsymbol{x})
\end{aligned}$$

The value returned by Eval is

$$\begin{aligned}
&\ell_{\mathsf{init}} + \boldsymbol{e}^{\top}_{(1,1,\mathbf{0}_S,1)} \sum_{t=1}^{T+1} (\mathbf{M}_{N,S}(\boldsymbol{x}))^{t-1} \boldsymbol{\ell}_t \\
&= (\beta + \boldsymbol{e}^{\top}_{(1,1,\mathbf{0}_S,1)} \boldsymbol{r}_0) + (-\boldsymbol{e}^{\top}_{(1,1,\mathbf{0}_S,1)} \boldsymbol{r}_0 + zM|_{N,T,S}(\boldsymbol{x})) \\
&= \beta + zM|_{N,T,S}(\boldsymbol{x}).
\end{aligned}$$

Therefore, the scheme is correct. Moreover, Eval is linear in the labels, as seen from the formula of Eval.

Theorem 2 [62] *The above construction of* AKGS *is piecewise secure. More precisely, the label functions are ordered as* $L_{\mathsf{init}}, (L_{1,\theta})_{\theta\in\mathcal{C}_{M,N,S}}, (L_{2,\theta})_{\theta\in\mathcal{C}_{M,N,S}}, \ldots, (L_{T+1,\theta})_{\theta\in\mathcal{C}_{M,N,S}}$, *the randomness is ordered as* $\boldsymbol{r}_0, \boldsymbol{r}_1, \ldots, \boldsymbol{r}_T$, *and the randomizer of* $L_{t,\theta}$ *is* $\boldsymbol{r}_{t-1}[\theta]$. *For each* $t \in [T+1]$, *the ordering of the components in* $(L_{i,\theta})_{\theta\in\mathcal{C}_{M,N,S}}$ *and* $\boldsymbol{r}_{t-1}$ *can be arbitrary, as long as the two are consistent.*

An exercise of algebra We note that the above construction of AKGS for the function class $\mathcal{F}$ requires to sample $\boldsymbol{r} \leftarrow \mathbb{Z}_p^{[0,T]\times\mathcal{C}_{M,N,S}}$. We will use "structured" element $\boldsymbol{r} = \boldsymbol{r}_{\boldsymbol{x}} \otimes \boldsymbol{r}_f$ for $\boldsymbol{r}_{\boldsymbol{x}} \leftarrow \mathbb{Z}_p^{[0,T]\times[N]\times[S]\times\{0,1\}^S}$ and $\boldsymbol{r}_f \leftarrow \mathbb{Z}_p^Q$ as the randomness for the AKGS garbling. We show that $\mathbf{M}_{N,S}(\boldsymbol{x})\boldsymbol{r}_t$ (a central part of the label functions) can be expressed as a *bilinear* function of $\boldsymbol{x}, \boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{x} \otimes \boldsymbol{r}_{\boldsymbol{x}}$ (known at encryption time) and $\mathbf{M}_\tau \boldsymbol{r}_f, \boldsymbol{r}_f$'s (known at key generation time), and hence can be computed as the inner products of vectors depending on these two groups of variables separately.

By our choice of randomness, $\boldsymbol{r}_t = \mathbf{r}[t, \llcorner, \llcorner, \llcorner, \llcorner]$ is a block vector with each block being a multiple of $\boldsymbol{r}_f$. More precisely, $\boldsymbol{r}_t[i, j, \boldsymbol{W}, \llcorner] = \boldsymbol{r}_{\boldsymbol{x}}[(k, t, i, j, \boldsymbol{W})]\boldsymbol{r}_f$. We compute each block of the product $\mathbf{M}_{N,S}(\boldsymbol{x})\boldsymbol{r}_t$:

$$\begin{aligned}
&(\mathbf{M}_{N,S}(\boldsymbol{x})\boldsymbol{r}_t)[(i, j, \boldsymbol{W}, \llcorner)] \\
\left(\begin{matrix}\text{row } r \text{ of } AB \text{ is} \\ \text{row } r \text{ of } A \text{ times } B\end{matrix}\right) \quad &= \mathbf{M}_{N,S}(\boldsymbol{x})[(i, j, \boldsymbol{W}, \llcorner), (\llcorner, \llcorner, \llcorner, \llcorner)]\boldsymbol{r}_t \\
\left(\begin{matrix}\text{block matrix} \\ \text{multiplication}\end{matrix}\right) \quad &= \sum_{\substack{i'\in[N], j'\in[S] \\ \boldsymbol{W}'\in\{0,1\}^S}} \mathbf{M}_{N,S}(\boldsymbol{x})[(i, j, \boldsymbol{W}, \llcorner), (i', j', \boldsymbol{W}', \llcorner)]\boldsymbol{r}_t[(i', j', \boldsymbol{W}', \llcorner)] \\
&= \sum_{\substack{i'\in[N], j'\in[S] \\ \boldsymbol{W}'\in\{0,1\}^S}} \mathbf{M}_{N,S}(\boldsymbol{x})[(i, j, \boldsymbol{W}, \llcorner), (i', j', \boldsymbol{W}', \llcorner)]\boldsymbol{r}_{\boldsymbol{x}}[(t, i', j', \boldsymbol{W}')]\boldsymbol{r}_f
\end{aligned}$$

Recall that in $\mathbf{M}_{N,S}(\boldsymbol{x})[(i, j, \boldsymbol{W}, \llcorner\!\lrcorner), (\llcorner\!\lrcorner, \llcorner\!\lrcorner, \llcorner\!\lrcorner, \llcorner\!\lrcorner)]$, each transition block appears at most once, and the other $Q \times Q$ blocks are $\mathbf{0}$. More specifically, $\mathbf{M}_{x,w,w',\Delta i,\Delta j}$ appears at $\mathbf{M}_{N,S}(\boldsymbol{x})[(i, j, \boldsymbol{W}, \llcorner\!\lrcorner), (i', j', \boldsymbol{W}', \llcorner\!\lrcorner)]$ if $x = \boldsymbol{x}[i]$, $w = \boldsymbol{W}[j]$, $\Delta i = i' - i$, $\Delta j = j' - j$, and $\boldsymbol{W}'$ is $\boldsymbol{W}$ with j-th entry changed to w'. Therefore, we have

$$\begin{aligned}
&(\mathbf{M}_{N,S}(\boldsymbol{x})\boldsymbol{r}_t)[(i, j, \boldsymbol{W}, \llcorner\!\lrcorner)] \\
&= \sum_{\substack{w' \in \{0,1\} \\ \Delta i, \Delta j \in \{0, \pm 1\} \\ i+\Delta i \in [N], j+\Delta j \in [S]}} \mathbf{M}_{\boldsymbol{x}[i],\boldsymbol{W}[j],w',\Delta i,\Delta j} \boldsymbol{r}_{\boldsymbol{x}}[(t, i+\Delta i, j+\Delta j, \boldsymbol{W}')]\boldsymbol{r}_f \\
&= \sum_{\substack{x,w,w' \in \{0,1\} \\ \Delta i, \Delta j \in \{0, \pm 1\}}} \mathbf{M}_{x,w,w',\Delta i,\Delta j} \boldsymbol{r}_f \times \\
&\quad \begin{cases} \boldsymbol{r}_{\boldsymbol{x}}[(t, i+\Delta i, j+\Delta j, \boldsymbol{W}')], & \text{if } x = \boldsymbol{x}[i], i+\Delta i \in [N], \\ & \quad w = \boldsymbol{W}[j], j+\Delta j \in [S]; \\ 0, & \text{otherwise} \end{cases}
\end{aligned} \tag{2}$$

Here, $\boldsymbol{W}'[j] = w'$ and $\boldsymbol{W}'[j''] = \boldsymbol{W}[j'']$ for all $j'' \neq j$. Note that in the last summation formula, there are exactly 72 summands. Moreover, each summand is $\mathbf{M}_{x,w,w',\Delta i,\Delta j}\boldsymbol{r}_f$ (depending only on $\boldsymbol{r}_f$ and the transition blocks) multiplied by an entry in $\boldsymbol{r}_{\boldsymbol{x}}$ or 0 (depending only on $\boldsymbol{x}, \boldsymbol{r}_{\boldsymbol{x}}$). To simplify notations, we define *transition coefficients*:

Definition 10 Let $\mathcal{T} = \{0,1\}^3 \times \{0, \pm 1\}^2$ be the set of transition types. For all $\tau = (x, w, w', \Delta i, \Delta j) \in \mathcal{T}$, $N, T, S \geq 1$, and $\boldsymbol{x} \in \{0,1\}^N$, $t \in [T]$, $i \in [N]$, $j \in [S]$, $\boldsymbol{W} \in \{0,1\}^S$, $\boldsymbol{r}_{\boldsymbol{x}} \in \mathbb{Z}_p^{[0,T]\times[N]\times[S]\times\{0,1\}^S}$, define the transition coefficient as

$$\begin{aligned}
&c_{x,w,w',\Delta i,\Delta j}(\boldsymbol{x}; t, i, j, \boldsymbol{W}; \boldsymbol{r}_{\boldsymbol{x}}) \\
&= \begin{cases} \boldsymbol{r}_{\boldsymbol{x}}[(t, i+\Delta i, j+\Delta j, \boldsymbol{W}')], & \text{if } x = \boldsymbol{x}[i], i+\Delta i \in [N], \\ & \quad w = \boldsymbol{W}[j], j+\Delta j \in [S]; \\ 0, & \text{otherwise} \end{cases}
\end{aligned}$$

where $\boldsymbol{W}' \in \{0,1\}^S$, $\boldsymbol{W}'[j] = w'$, and $\boldsymbol{W}'[j''] = \boldsymbol{W}[j'']$ for all $j'' \neq j$.

With the above definition, Eq. (2) can be restated as

$$(\mathbf{M}_{N,S}(\boldsymbol{x})\boldsymbol{r}_t)[(i, j, \boldsymbol{W}, \llcorner\!\lrcorner)] = \sum_{\tau \in \mathcal{T}} c_\tau(\boldsymbol{x}, t, i, j, \boldsymbol{W}; \boldsymbol{r}_{\boldsymbol{x}})\mathbf{M}_\tau \boldsymbol{r}_f. \tag{3}$$

5 (1-SK, 1-CT, 1-slot)-FE for unbounded AWS in L

In this section, we build a *secret-key*, 1-slot FE scheme for the *unbounded* attribute-weighted sum functionality in L. At a high level, the scheme satisfies the following properties:

– The setup is *independent* of any parameters, other than the security parameter λ. Specifically, the *length* of vectors and attributes, *number* of Turing machines and their *sizes* are not fixed a-priori during setup. These parameters are flexible and can be chosen at the time of key generation or encryption.

- A secret key is associated with a tuple $(\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}})$, where $\boldsymbol{M} = (M_k)_{k \in \mathcal{I}_{\boldsymbol{M}}}$ is a tuple of Turing machines with indices k from an index set $\mathcal{I}_{\boldsymbol{M}}$. For each $k \in \mathcal{I}_{\boldsymbol{M}}$, $M_k \in \mathsf{L}$, i.e., M_k is represented by a deterministic log-space bounded Turing machine (with an arbitrary number of states).
- Each ciphertext encodes a tuple of public–private attributes $(\boldsymbol{x}, \boldsymbol{z})$ of lengths N and n respectively. The runtime T and space bound S for all the machines in $\boldsymbol{M}$ are associated with $\boldsymbol{x}$ which is the input of each machine M_k.
- Finally, decrypting a ciphertext $\mathsf{CT}_{\boldsymbol{x}}$ that encodes $(\boldsymbol{x}, \boldsymbol{z})$ with a secret key $\mathsf{SK}_{\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}}}$ that is tied to $(\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}})$ reveals the value $\sum_{k \in \mathcal{I}_{\boldsymbol{M}}} \boldsymbol{z}[k] \cdot M_k(\boldsymbol{x})$ whenever $\mathcal{I}_{\boldsymbol{M}} \subseteq [n]$.

We build an FE scheme for the functionality sketched above (also described in Definition 2) and prove it to be simulation secure against a *single* ciphertext and secret key query, where the key can be asked either before or after the ciphertext query. Accordingly, we denote the scheme as $\mathsf{SK\text{-}UAWS}^{\mathsf{L}}_{(1,1,1)} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, where the index $(1, 1, 1)$ represents in order the number of secret keys, ciphertexts and slots supported. Below, we list the ingredients for our scheme.

1. $\mathsf{IPFE} = (\mathsf{IPFE.Setup}, \mathsf{IPFE.KeyGen}, \mathsf{IPFE.Enc}, \mathsf{IPFE.Dec})$: a *secret-key*, *function-hiding* IPFE based on G, where $\mathsf{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathrm{T}, g_1, g_2, e)$ is pairing group tuple of prime order p. We can instantiate this from [62].
2. $\mathsf{AKGS} = (\mathsf{Garble}, \mathsf{Eval})$: a special piecewise-secure AKGS for the function class $\mathcal{M} = \{M|_{N,T,S} : \mathbb{Z}_p^N \to \mathbb{Z}_p \mid M \in \mathsf{TM}, N, T, S \geq 1, p \text{ prime}\}$
describing the set of time/space bounded Turing machines. In our construction, the Garble algorithm would run implicitly under the hood of IPFE and thus, it is not invoked directly in the scheme.

5.1 The construction

We are now ready to describe the $\mathsf{SK\text{-}UAWS}^{\mathsf{L}}_{(1,1,1)} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$.

$\mathsf{Setup}(1^\lambda)$: On input the security parameter, fix a prime integer $p \in \mathbb{N}$ and define the slots for two IPFE master secret keys as follows:

$$\begin{aligned} \mathcal{S}_{\mathsf{1\text{-}UAWS}} =& \{\mathsf{index}_1, \mathsf{index}_2, \mathsf{init}, \mathsf{rand}, \mathsf{rand}^{\mathsf{temp}}, \mathsf{rand}^{\mathsf{comp}}, \mathsf{rand}^{\mathsf{temp,comp}}, \mathsf{acc}, \mathsf{sim}, \mathsf{sim}^{\mathsf{temp}}, \mathsf{sim}^{\mathsf{comp}}\} \\ & \cup \{\mathsf{tb}_\tau, \mathsf{tb}_\tau^{\mathsf{temp}}, \mathsf{tb}_\tau^{\mathsf{comp}}, \mathsf{tb}_\tau^{\mathsf{temp,comp}} \mid \tau \in \mathcal{T}\}, \quad (\mathcal{T} \text{ is defined in Definition 10}) \\ \widetilde{\mathcal{S}}_{\mathsf{1\text{-}UAWS}} =& \{\mathsf{index}_1, \mathsf{index}_2, \mathsf{init}, \mathsf{rand}, \mathsf{rand}^{\mathsf{temp}}, \mathsf{rand}^{\mathsf{temp,comp}}, \mathsf{acc}, \mathsf{acc}^{\mathsf{temp}}, \mathsf{sim}, \mathsf{sim}^{\mathsf{temp}}\}. \end{aligned}$$

Finally, it returns $\mathsf{MSK} = (\mathsf{IPFE.MSK}, \mathsf{IPFE}.\widetilde{\mathsf{MSK}})$.

rlap$\mathsf{KeyGen}(\mathsf{MSK}, (\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}}))$: On input the master secret key $\mathsf{MSK} = (\mathsf{IPFE.MSK}, \mathsf{IPFE}.\widetilde{\mathsf{MSK}})$ and a function tuple $\boldsymbol{M} = (M_k)_{k \in \mathcal{I}_{\boldsymbol{M}}}$ indexed w.r.t. an index set $\mathcal{I}_{\boldsymbol{M}} \subset \mathbb{N}$ of arbitrary size , parse $M_k = (Q_k, \boldsymbol{y}_k, \delta_k) \in \mathsf{TM}\ \forall k \in \mathcal{I}_{\boldsymbol{M}}$ and sample the set of elements

$$\left\{\beta_k \leftarrow \mathbb{Z}_p \mid \sum_k \beta_k = 0 \mod p\right\}_{k \in \mathcal{I}_{\boldsymbol{M}}}$$

For all $k \in \mathcal{I}_{\boldsymbol{M}}$, do the following:

1. For $M_k = (Q_k, \boldsymbol{y}_k, \delta_k)$, compute its transition blocks $\mathbf{M}_{k,\tau} \in \{0, 1\}^{Q_k \times Q_k}, \forall \tau \in \mathcal{T}$.
2. Sample independent random vectors $\boldsymbol{r}_{k,f} \leftarrow \mathbb{Z}_p^{Q_k}$ and a random element $\pi_k \in \mathbb{Z}_p$.
3. For the following vector $\boldsymbol{v}_{k,\mathsf{init}}$, compute a secret key $\mathsf{IPFE.SK}_{k,\mathsf{init}} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{k,\mathsf{init}}]\!]_2)$:

vector	index_1	index_2	init	rand	acc	tb_τ	the other indices
$\boldsymbol{v}_{k,\text{init}}$	π_k	$k \cdot \pi_k$	$\boldsymbol{r}_{k,f}[1]$	0	β_k	0	0

4. For each $q \in [Q_k]$, compute the following secret keys

$$\mathsf{IPFE.SK}_{k,q} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{k,q}]\!]_2) \quad \text{and}$$
$$\widetilde{\mathsf{IPFE.SK}}_{k,q} \leftarrow \mathsf{IPFE.KeyGen}(\widetilde{\mathsf{IPFE.MSK}}, [\![\widetilde{\boldsymbol{v}}_{k,q}]\!]_2),$$

where the vectors $\boldsymbol{v}_{k,q}$, $\widetilde{\boldsymbol{v}}_{k,q}$ are defined as follows:

vector	index_1	index_2	init	rand	acc	tb_τ	the other indices
$\boldsymbol{v}_{k,q}$	π_k	$k \cdot \pi_k$	0	$-\boldsymbol{r}_{k,f}[q]$	0	$\left(\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f}\right)[q]$	0

vector	index_1	index_2	rand	acc	the other indices
$\widetilde{\boldsymbol{v}}_{k,q}$	π_k	$k \cdot \pi_k$	$-\boldsymbol{r}_{k,f}[q]$	$\boldsymbol{y}_k[q]$	0

Finally, it returns the secret key as

$$\mathsf{SK}_{(\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}})} = \left((\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}}), \left\{\mathsf{IPFE.SK}_{k,\text{init}}, \{\mathsf{IPFE.SK}_{k,q}, \widetilde{\mathsf{IPFE.SK}}_{k,q}\}_{q\in[Q_k]}\right\}_{k\in\mathcal{I}_{\boldsymbol{M}}}\right).$$

$\mathsf{Enc}(\mathsf{MSK}, (\boldsymbol{x}, 1^T, 1^{2^S}), \boldsymbol{z})$: On input the master secret key $\mathsf{MSK} = (\mathsf{IPFE.MSK}, \widetilde{\mathsf{IPFE.MSK}})$, a public attribute $\boldsymbol{x} \in \{0, 1\}^N$ for some arbitrary $N \geq 1$ with time and space complexity bounds given by $T, S \geq 1$ (as $1^T, 1^{2^S}$) respectively, and the private attribute $\boldsymbol{z} \in \mathbb{Z}_p^n$ for some arbitrary $n \geq 1$, it does the following:

1. Sample a random vector $\boldsymbol{r}_{\boldsymbol{x}} \leftarrow \mathbb{Z}_p^{[0,T]\times[N]\times[S]\times\{0,1\}^S}$.
2. For each $k \in [n]$, do the following:

   (a) Sample a random element $\rho_k \leftarrow \mathbb{Z}_p$.
   (b) Compute a ciphertext $\mathsf{IPFE.CT}_{k,\text{init}} \leftarrow \mathsf{IPFE.Enc}(\mathsf{IPFE.MSK}, [\![\boldsymbol{u}_{k,\text{init}}]\!]_1)$ for the vector $\boldsymbol{u}_{k,\text{init}}$:

vector	index_1	index_2	init	rand	acc	tb_τ	the other indices
$\boldsymbol{u}_{k,\text{init}}$	$-k \cdot \rho_k$	ρ_k	$\boldsymbol{r}_{\boldsymbol{x}}[(0, 1, 1, \boldsymbol{0}_S)]$	0	1	0	0

   (c) For all $t \in [T]$, $i \in [N]$, $j \in [S]$, $\boldsymbol{W} \in \{0, 1\}^S$, do the following:
      (i) Compute the transition coefficients $c_\tau(\boldsymbol{x}; t, i, j, \boldsymbol{W}; \boldsymbol{r}_{\boldsymbol{x}}), \forall \tau \in \mathcal{T}$ using $\boldsymbol{r}_{\boldsymbol{x}}$.
      (ii) Compute the ciphertext $\mathsf{IPFE.CT}_{k,t,i,j,\boldsymbol{W}} \leftarrow \mathsf{IPFE.Enc}(\mathsf{IPFE.MSK}, [\![\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}]\!]_1)$ for the vector $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$:

vector	index_1	index_2	init	rand	acc	tb_τ	the other indices
$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$	$-k \cdot \rho_k$	ρ_k	0	$\boldsymbol{r}_{\boldsymbol{x}}[(t-1, i, j, \boldsymbol{W})]$	0	$c_\tau(\boldsymbol{x}; t, i, j, \boldsymbol{W}; \boldsymbol{r}_{\boldsymbol{x}})$	0

   (d) For $t = T + 1$, compute the ciphertext $\widetilde{\mathsf{IPFE.CT}}_{k,T+1,i,j,\boldsymbol{W}} \leftarrow \widetilde{\mathsf{IPFE.Enc}}(\widetilde{\mathsf{IPFE.MSK}}, [\![\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}]\!]_1)$ for the vector $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$:

vector	index_1	index_2	rand	acc	the other indices
$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	$-k \cdot \rho_k$	ρ_k	$\boldsymbol{r}_{\boldsymbol{x}}[(T, i, j, \boldsymbol{W})]$	$\boldsymbol{z}[k]$	0

3. Finally, it returns the ciphertext as

$$\mathsf{CT}_{(\boldsymbol{x},T,S)} = \Bigg((\boldsymbol{x}, T, S)\,, \Big\{ \mathsf{IPFE.CT}_{k,\mathsf{init}}, \{\mathsf{IPFE.CT}_{k,t,i,j,\boldsymbol{W}}\}_{t\in[T]}, \widetilde{\mathsf{IPFE.CT}}_{k,T+1,i,j,\boldsymbol{W}} \Big\}_{k\in[n],i\in[N],j\in[S],\boldsymbol{W}\in\{0,1\}^S} \Bigg).$$

$\mathsf{Dec}(\mathsf{SK}_{(\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}})}, \mathsf{CT}_{(\boldsymbol{x},T,S)})$: On input a secret key $\mathsf{SK}_{(\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}})}$ and a ciphertext $\mathsf{CT}_{(\boldsymbol{x},T,S)}$, do the following:

1. Parse $\mathsf{SK}_{(\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}})}$ and $\mathsf{CT}_{(\boldsymbol{x},T,S)}$ as follows:

$$\mathsf{SK}_{(\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}})} = \Big(\big((M_k)_{k\in\mathcal{I}_{\boldsymbol{M}}}, \mathcal{I}_{\boldsymbol{M}}\big)\,, \Big\{ \mathsf{IPFE.SK}_{k,\mathsf{init}}, \big\{\mathsf{IPFE.SK}_{k,q}, \widetilde{\mathsf{IPFE.SK}}_{k,q}\}_{q\in[Q_k]} \Big\}_{k\in\mathcal{I}_{\boldsymbol{M}}} \Big),$$
$$M_k = (Q_k, \boldsymbol{y}_k, \delta_k),$$
$$\mathsf{CT}_{(\boldsymbol{x},T,S)} = \Big((\boldsymbol{x}, T, S)\,, \Big\{ \mathsf{IPFE.CT}_{k,\mathsf{init}}, \{\mathsf{IPFE.CT}_{k,t,i,j,\boldsymbol{W}}\}_{t\in[T]}, \widetilde{\mathsf{IPFE.CT}}_{k,T+1,i,j,\boldsymbol{W}} \Big\}_{k\in[n],i\in[N],j\in[S],\boldsymbol{W}\in\{0,1\}^S} \Big), \boldsymbol{x} \in \{0,1\}^N.$$

2. Output $\perp$, if $\mathcal{I}_{\boldsymbol{M}} \not\subseteq [n]$. Else, select the sequence of ciphertexts for the indices $k \in \mathcal{I}_{\boldsymbol{M}}$ as

$$\mathsf{CT}_{(\boldsymbol{x},T,S)} = \Bigg((\boldsymbol{x}, T, S)\,, \Big\{ \mathsf{IPFE.CT}_{k,\mathsf{init}}, \{\mathsf{IPFE.CT}_{k,t,i,j,\boldsymbol{W}}\}_{t\in[T]}, \widetilde{\mathsf{IPFE.CT}}_{k,T+1,i,j,\boldsymbol{W}} \Big\}_{k\in\mathcal{I}_{\boldsymbol{M}},i\in[N],j\in[S],\boldsymbol{W}\in\{0,1\}^S} \Bigg)$$

3. Recall that $\forall k \in \mathcal{I}_{\boldsymbol{M}}, \mathcal{C}_{M_k,N,S} = [N] \times [S] \times \{0,1\}^S \times [Q_k]$, and that we denote any element in it as $\theta_k = (i, j, \boldsymbol{W}, q) \in \mathcal{C}_{M_k,N,S}$ where the only component in the tuple θ_k depending on k is $q \in [Q_k]$[2]. Invoke the IPFE decryption to compute all label values as:

$$\forall k \in \mathcal{I}_{\boldsymbol{M}} : [\![\ell_{k,\mathsf{init}}]\!]_{\mathrm{T}} = \mathsf{IPFE.Dec}(\mathsf{IPFE.SK}_{k,\mathsf{init}}, \mathsf{IPFE.CT}_{k,\mathsf{init}})$$

$$\forall k \in \mathcal{I}_{\boldsymbol{M}}, t \in [T], \theta_k = (i, j, \boldsymbol{W}, q) \in \mathcal{C}_{M_k,N,S} :$$
$$[\![\ell_{k,t,\theta_k}]\!]_{\mathrm{T}} = \mathsf{IPFE.Dec}(\mathsf{IPFE.SK}_{k,q}, \mathsf{IPFE.CT}_{k,t,i,j,\boldsymbol{W}})$$

$$\forall k \in \mathcal{I}_{\boldsymbol{M}}, \theta_k = (i, j, \boldsymbol{W}, q) \in \mathcal{C}_{M_k,N,S} :$$
$$[\![\ell_{k,T+1,\theta_k}]\!]_{\mathrm{T}} = \mathsf{IPFE.Dec}(\widetilde{\mathsf{IPFE.SK}}_{k,q}, \widetilde{\mathsf{IPFE.CT}}_{k,T+1,i,j,\boldsymbol{W}})$$

4. Next, invoke the AKGS evaluation and obtain the combined value

$$[\![\mu]\!]_{\mathrm{T}} = \prod_{k\in\mathcal{I}_{\boldsymbol{M}}} \mathsf{Eval}\Big(\big(M_k, 1^N, 1^T, 1^{2^S}, p\big), \boldsymbol{x}, [\![\ell_{k,\mathsf{init}}]\!]_{\mathrm{T}}, \Big\{ [\![\ell_{k,t,\theta_k}]\!]_{\mathrm{T}} \Big\}_{t\in[T+1],\theta_k\in\mathcal{C}_{M_k,N,S}} \Big)$$

5. Finally, it returns $\mu = \mathsf{DLog}_{g_{\mathrm{T}}}([\![\mu]\!]_{\mathrm{T}})$, where $g_{\mathrm{T}} = e(g_1, g_2)$. Similar to [8], we assume that the desired attribute-weighted sum lies within a specified polynomial-sized domain so that discrete logarithm can be solved via brute-force.

Correctness Correctness follows from that of IPFE and AKGS. The first step is to observe that all the AKGS label values are correctly computed as functions of the input $\boldsymbol{x}$. This holds by the

[2] For simplicity of notations, we enumerate the states of each M_k as $1, \ldots, q$, i.e., $[Q_k] = [Q]$ for some $Q \in \mathbb{N}$.

correctness of IPFE and AKGS encoding of the iterated matrix-vector product representing any TM computation. The next (and final) correctness follows from the linearity of AKGS.Eval.

In more detail, for all $k \in \mathcal{I}_{\boldsymbol{M}}, \theta_k = (i, j, \boldsymbol{W}, q) \in \mathcal{C}_{M_k,N,S}$, let $L_{k,\mathsf{init}}, L_{k,t,\theta_k}$ be the label functions corresponding to the AKGS garbling of $M_k = (Q_k, \boldsymbol{y}_k, \delta_k)$. By the definitions of vectors $\boldsymbol{v}_{k,\mathsf{init}}, \boldsymbol{u}_{\mathsf{init}}$ and the correctness of IPFE, we have

$$\begin{aligned}\ell_{k,\mathsf{init}} &= (-k\rho_k\pi_k + k\pi_k\rho_k) + \boldsymbol{r}_{\boldsymbol{x}}[(0,1,1,\boldsymbol{0}_S)]\boldsymbol{r}_{k,f}[1] + \beta_k\\ &= \boldsymbol{r}_0[(1,1,\boldsymbol{0}_S,1)] + \beta_k = \boldsymbol{e}^T_{(1,1,\boldsymbol{0}_S,1)}\boldsymbol{r}_0 + \beta_k = L_{k,\mathsf{init}}(\boldsymbol{x}).\end{aligned}$$

Next, $\forall k \in \mathcal{I}_{\boldsymbol{M}}, t \in [T], q \in [Q_k]$, the structures of $\boldsymbol{v}_{k,q}, \boldsymbol{u}_{t,i,j,\boldsymbol{W}}$ and the correctness of IPFE yields

$$\begin{aligned}&\ell_{k,t,i,j,\boldsymbol{W},q}\\ &\quad= (-k\rho_k\pi_k + k\pi_k\rho_k) - \boldsymbol{r}_{\boldsymbol{x}}[(t-1,i,j,\boldsymbol{W})]\boldsymbol{r}_{k,f}[q]\\ &\quad+ \sum_{\tau\in\mathcal{T}} c_\tau(\boldsymbol{x}; t,i,j,\boldsymbol{W}; \boldsymbol{r}_{\boldsymbol{x}})(\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q]\\ &\quad= -\boldsymbol{r}_{t-1}[(i,j,\boldsymbol{W},q)] + \sum_{\tau\in\mathcal{T}} c_\tau(\boldsymbol{x}; t,i,j,\boldsymbol{W}; \boldsymbol{r}_{\boldsymbol{x}})(\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q] = L_{k,t,i,j,\boldsymbol{W},q}(\boldsymbol{x})\end{aligned}$$

Finally, $\forall k \in \mathcal{I}_{\boldsymbol{M}}, q \in [Q_k]$, the vectors $\widetilde{\boldsymbol{v}}_{k,q}, \widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$ and the $\widetilde{\mathsf{IPFE}}$ correctness again yields

$$\begin{aligned}\ell_{k,T+1,i,j,\boldsymbol{W},q} &= (-k\rho_k\pi_k + k\pi_k\rho_k) - \boldsymbol{r}_{\boldsymbol{x}}[(T,i,j,\boldsymbol{W})]\boldsymbol{r}_{k,f}[q] + \boldsymbol{z}[k]\boldsymbol{y}_k[q]\\ &= -\boldsymbol{r}_T[(i,j,\boldsymbol{W},q)] + \boldsymbol{z}[k]\left(1_{[N]\times[S]\times\{0,1\}^S}\otimes \boldsymbol{y}_k\right)[(i,j,\boldsymbol{W},q)]\\ &= L_{k,T+1,i,j,\boldsymbol{W},q}(\boldsymbol{x}).\end{aligned}$$

The above label values are computed in the exponent of the target group $\mathbb{G}_{\mathrm{T}}$. Once all these are generated correctly, the linearity of Eval implies that the garbling can be evaluated in the exponent of $\mathbb{G}_{\mathrm{T}}$. Thus, this yields

$$\begin{aligned}[\![\mu]\!]_{\mathrm{T}} &= \prod_{k\in\mathcal{I}_{\boldsymbol{M}}} \mathsf{Eval}\left(\left(M_k, 1^N, 1^T, 1^{2^S}, p\right), \boldsymbol{x}, [\![\ell_{k,\mathsf{init}}]\!]_{\mathrm{T}}, \left\{[\![\ell_{k,t,\theta_k}]\!]_{\mathrm{T}}\right\}_{t\in[T+1],\theta_k\in\mathcal{C}_{M_k,N,S}}\right)\\ &= [\![\sum_{k\in\mathcal{I}_{\boldsymbol{M}}} \mathsf{Eval}((M_k, 1^N, 1^T, 1^{2^S}, p), \boldsymbol{x}, \ell_{k,\mathsf{init}}, \{\ell_{k,t,\theta_k}\}_{t\in[T+1],\theta_k\in\mathcal{C}_{M_k,N,S}})]\!]_{\mathrm{T}}\\ &= [\![\sum_{k\in\mathcal{I}_{\boldsymbol{M}}} (\boldsymbol{z}[k]\cdot M_k|_{N,T,S}(\boldsymbol{x}) + \beta_k)]\!]_{\mathrm{T}} = [\![\sum_{k\in\mathcal{I}_{\boldsymbol{M}}} \boldsymbol{z}[k]\cdot M_k|_{N,T,S}(\boldsymbol{x})]\!]_{\mathrm{T}} = [\![\boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{z}]\!]_{\mathrm{T}}\end{aligned}$$

5.2 Security analysis

We describe the simulator of our (1-SK, 1-CT, 1-Slot)-FE for UAWS. The simulated setup Setup^* operates exactly the same way as the honest setup works. The simulated master secret key is $\mathsf{MSK}^* = (\mathsf{IPFE.MSK}, \widetilde{\mathsf{IPFE}}.\mathsf{MSK})$. The simulated key generation algorithm KeyGen_0^* also works in the same fashion as the honest key generation proceeds. We now describe the simulated encryption Enc^* and the simulated key generation KeyGen_1^* below.

$\mathsf{Enc}^*(\mathsf{MSK}^*, (\boldsymbol{x}, 1^T, 1^{2^S}), (\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}}, \boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{z}), n)$: On input the simulated master secret key MSK^*, the challenge public attribute $\boldsymbol{x}$ with associated parameters $T, 2^S$ in unary, (if there is a secret key query before the challenge ciphertext is generated then) the secret key-functional value tuple $(\boldsymbol{M} = (M_k)_{k\in\mathcal{I}_{\boldsymbol{M}}}, \mathcal{I}_{\boldsymbol{M}}, \boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{z} = \sum_{k\in\mathcal{I}_{\boldsymbol{M}}} M_k(\boldsymbol{x})\boldsymbol{z}[k])$ with $\mathcal{I}_{\boldsymbol{M}} \subseteq [n]$ and the length of the private attribute n, the encryption proceeds as follows:

1. It samples a dummy vector $\boldsymbol{d} \leftarrow \mathbb{Z}_p^n$ such that

$$\boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{z} = \boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{d} = \sum_{k \in [n]} M_k(\boldsymbol{x})\boldsymbol{d}[k].$$

Note that, it can always set $M_k(\boldsymbol{x}) = 0$ for $k \notin [n] \setminus \mathcal{I}_{\boldsymbol{M}}$. If there is no secret key query before the challenge ciphertext then it chooses a random vector $\boldsymbol{v} \in \mathbb{Z}_p^n$ in place of $\boldsymbol{d}$.

2. Sample a random vector $\boldsymbol{r_x} \leftarrow \mathbb{Z}_p^{[0,T]\times[N]\times[S]\times\{0,1\}^S}$ and $\boldsymbol{s_x} \leftarrow \mathbb{Z}_p^{[T+1]\times[N]\times[S]\times\{0,1\}^S}$.
3. For each $k \in [n]$, do the following:
   (a) Sample a random element $\rho_k \leftarrow \mathbb{Z}_p$.
   (b) Compute a ciphertext $\mathsf{IPFE.CT}_{k,\mathsf{init}} \leftarrow \mathsf{IPFE.Enc}(\mathsf{IPFE.MSK}, [\![\boldsymbol{u}_{k,\mathsf{init}}]\!]_1)$ for the vector $\boldsymbol{u}_{k,\mathsf{init}}$:

vector	index$_1$	index$_2$	init	acc	sim	the other indices
$\boldsymbol{u}_{k,\mathsf{init}}$	$-k \cdot \rho_k$	ρ_k	$\boldsymbol{r_x}[(0, 1, 1, \boldsymbol{0}_S)]$	1	1	0

   (c) For all $t \in [T]$, $i \in [N]$, $j \in [S]$, $\boldsymbol{W} \in \{0, 1\}^S$, do the following:
      (i) Compute the coefficients $c_\tau(\boldsymbol{x}; t, i, j, \boldsymbol{W}; \boldsymbol{r_x})$, $\forall \tau \in \mathcal{T}$ using $\boldsymbol{r_x}$.
      (ii) Compute the ciphertext $\mathsf{IPFE.CT}_{k,t,i,j,\boldsymbol{W}} \leftarrow \mathsf{IPFE.Enc}(\mathsf{IPFE.MSK}, [\![\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}]\!]_1)$ for the vector $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$:

vector	index$_1$	index$_2$	rand	tb$_\tau$	sim	the other indices
$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$	$-k \cdot \rho_k$	ρ_k	$\boldsymbol{r_x}[(t-1, i, j, \boldsymbol{W})]$	$c_\tau(\boldsymbol{x}; t, i, j, \boldsymbol{W}; \boldsymbol{r_x})$	$\boldsymbol{s_x}[(t, i, j, \boldsymbol{W})]$	0

   (d) For $t = T+1$, compute $\widetilde{\mathsf{IPFE.CT}}_{k,T+1,i,j,\boldsymbol{W}} \leftarrow \mathsf{IPFE.Enc}(\widetilde{\mathsf{IPFE.MSK}}, [\![\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}]\!]_1)$ for the vector $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$:

vector	index$_1$	index$_2$	rand	acc	sim	the other indices
$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	$-k \cdot \rho_k$	ρ_k	$\boldsymbol{r_x}[(T, i, j, \boldsymbol{W})]$	$\boldsymbol{v}[k]$ or $\boldsymbol{d}[k]$	$\boldsymbol{s_x}[(T+1, i, j, \boldsymbol{W})]$	0

4. Finally, it returns the ciphertext as

$$\mathsf{CT}_{(\boldsymbol{x},T,S)} = \left((\boldsymbol{x}, T, S), \left\{ \mathsf{IPFE.CT}_{k,\mathsf{init}}, \{\mathsf{IPFE.CT}_{k,t,i,j,\boldsymbol{W}}\}_{t\in[T]}, \widetilde{\mathsf{IPFE.CT}}_{k,T+1,i,j,\boldsymbol{W}} \right\}_{k\in[n], i\in[N], j\in[S], \boldsymbol{W}\in\{0,1\}^S} \right).$$

$\mathsf{KeyGen}_1^*(\mathsf{MSK}^*, (\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}}, \boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{z}))$: On input the master secret key MSK^* and the secret key-functional value tuple $(\boldsymbol{M} = (M_k)_{k\in\mathcal{I}_{\boldsymbol{M}}}, \mathcal{I}_{\boldsymbol{M}}, \boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{z} = \sum_{k\in\mathcal{I}_{\boldsymbol{M}}} M_k(\boldsymbol{x})\boldsymbol{z}[k])$ w.r.t. an index set $\mathcal{I}_{\boldsymbol{M}} \subset \mathbb{N}$, the key generation process works as follows:

1. It parses $M_k = (Q_k, \boldsymbol{y}_k, \delta_k) \in \mathsf{TM}\ \forall k \in \mathcal{I}_{\boldsymbol{M}}$ and sample elements $\beta'_k \in \mathbb{Z}_p$ for $k \in \mathcal{I}_{\boldsymbol{M}}$ as follows:

$$\begin{aligned} \text{if } \mathcal{I}_{\boldsymbol{M}} \subseteq [n]: &\quad \beta'_k \leftarrow \mathbb{Z}_p \text{ and } \textstyle\sum_k \beta'_k = 0 \mod p \\ \text{if } (\max \mathcal{I}_{\boldsymbol{M}} > n) \wedge (\min \mathcal{I}_{\boldsymbol{M}} \le n): &\quad \beta'_k \leftarrow \mathbb{Z}_p \end{aligned}$$

2. For $M_k = (Q_k, \boldsymbol{y}_k, \delta_k)$, compute transition blocks $\mathbf{M}_{k,\tau} \in \{0, 1\}^{Q_k \times Q_k}$, $\forall \tau \in \mathcal{T}_k$.

3. It reversely sample the label function values as

$$\ell_{1,\mathsf{init}} \leftarrow \mathsf{RevSamp}((M_1, 1^N, 1^T, 1^{2^S}), \boldsymbol{x}, \boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{z} + \beta'_1, (\ell_{k,t,\theta_k})_{t\in[T+1],\theta_k\in\mathcal{C}_{M_k,N,S}})$$
$$\ell_{k,\mathsf{init}} \leftarrow \mathsf{RevSamp}((M_k, 1^N, 1^T, 1^{2^S}), \boldsymbol{x}, \beta'_k, (\ell_{k,t,\theta_k})_{t\in[T+1],\theta_k\in\mathcal{C}_{M_k,N,S}}), \quad \text{for all } k > 1$$

where all the other label values $\ell_{k,t,i,j,\boldsymbol{W},q} = \boldsymbol{s_x}[(t,i,j,\boldsymbol{W})]\boldsymbol{s}_{k,f}[q]$ are simulated (and known to the simulator).

4. For the following vector $\boldsymbol{v}_{k,\mathsf{init}}$, compute a secret key $\mathsf{IPFE.SK}_{k,\mathsf{init}} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{k,\mathsf{init}}]\!]_2)$:

vector	index$_1$	index$_2$	sim	the other indices
$\boldsymbol{v}_{k,\mathsf{init}}$	π_k	$k\cdot\pi_k$	$\ell_{k,\mathsf{init}}$	0

5. For each $q \in [Q_k]$, compute the following secret keys

$$\mathsf{IPFE.SK}_{k,q} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{k,q}]\!]_2), \qquad \text{and}$$
$$\widetilde{\mathsf{IPFE.SK}}_{k,q} \leftarrow \mathsf{IPFE.KeyGen}(\widetilde{\mathsf{IPFE.MSK}}, [\![\widetilde{\boldsymbol{v}}_{k,q}]\!]_2),$$

where the vectors $\boldsymbol{v}_{k,q}, \widetilde{\boldsymbol{v}}_{k,q}$ are defined as follows:

vector	index$_1$	index$_2$	sim	the other indices
$\boldsymbol{v}_{k,q}$	π_k	$k\cdot\pi_k$	$\boldsymbol{s_x}[(t,i,j,\boldsymbol{W})]$	0

vector	index$_1$	index$_2$	sim	the other indices
$\widetilde{\boldsymbol{v}}_{k,q}$	k	$k\cdot\pi_k$	$\boldsymbol{s_x}[(T+1,i,j,\boldsymbol{W})]$	0

Note that, the random vector $\boldsymbol{s_x}$ has already been sampled during encryption.

Finally, it returns the simulated secret key as

$$\mathsf{SK}_{(\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}})} = \left((\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}}), \left\{\mathsf{IPFE.SK}_{k,\mathsf{init}}, \{\mathsf{IPFE.SK}_{k,q}, \widetilde{\mathsf{IPFE.SK}}_{k,q}\}_{q\in[Q_k]}\right\}_{k\in\mathcal{I}_{\boldsymbol{M}}}\right).$$

We will use the following lemmas in our security analysis.

Lemma 4 *Let* $\mathsf{IPFE} = (\mathit{Setup}, \mathit{KeyGen}, \mathit{Enc}, \mathit{Dec})$ *be a function hiding inner product encryption scheme. For any polynomial* $m = m(\lambda)$ *and* $n = n(\lambda)$ *with* $m > n$, *define the following vectors*

$$\begin{array}{lllllll} \pi_k, \rho_k, r_k, \widehat{r}_k \leftarrow \mathbb{Z}_p;\ \mathfrak{b} \leftarrow \{0,1\} \\ \boldsymbol{v}_k = (& \pi_k, & k\cdot\pi_k, & \boldsymbol{0}, & 0, & \boldsymbol{0}\ ) & \textit{for } k \in [n] \\ \boldsymbol{v}_k^{(\mathfrak{b})} = (& \pi_k, & k\cdot\pi_k, & \boldsymbol{0}, & r_k + \mathfrak{b}\cdot\widehat{r}_k, & \boldsymbol{0}\ ) & \textit{for } k \in [n+1,m] \\ \boldsymbol{u}_{k'} = (& -k'\cdot\rho_{k'}, & \rho_{k'}, & \boldsymbol{0}, & 1, & \boldsymbol{0}\ ) & \textit{for } k' \in [n] \end{array}$$

Then, for any $\mathsf{IPFE.MSK} \leftarrow \mathsf{IPFE.}\mathit{Setup}(1^\lambda, 1^*)$, *the distributions* $\{\{\mathsf{IPFE.SK}_k\}_{k\in[n]}, \{\mathsf{IPFE.SK}_k^{(\mathfrak{b})}\}_{k\in[n+1,m]}, \{\mathsf{IPFE.CT}_{k'}\}_{k'\in[n]}\}$ *for* $\mathfrak{b} \in \{0,1\}$ *are indistinguishable where*

$$\begin{array}{rll} \mathsf{IPFE.SK}_k & \leftarrow \mathsf{IPFE.}\mathit{KeyGen}(\mathsf{IPFE.MSK}, \boldsymbol{v}_k) & \textit{for } k \in [n] \\ \mathsf{IPFE.SK}_k^{(\mathfrak{b})} & \leftarrow \mathsf{IPFE.}\mathit{KeyGen}(\mathsf{IPFE.MSK}, \boldsymbol{v}_k^{(\mathfrak{b})}) & \textit{for } k \in [n+1,m] \\ \mathsf{IPFE.CT}_k & \leftarrow \mathsf{IPFE.}\mathit{Enc}(\mathsf{IPFE.MSK}, \boldsymbol{u}_{k'}) & \textit{for } k \in [n] \end{array}$$

Proof We prove this lemma by the transformation $\widehat{\pi}_k = \pi_k - \frac{\widehat{r}_k}{\rho_{k'}(k-k')}$ for $k \neq k'$. Note that $\widehat{\pi}_k$ is uniform over $\mathbb{Z}_p$ since $\pi_k \leftarrow \mathbb{Z}_p$. The lemma follows from the function hiding security of IPFE since

$$\begin{aligned} \boldsymbol{v}^{(0)} \cdot \boldsymbol{u}_{k'} &= \pi_k \rho_{k'} \cdot (k - k') + r_k \\ &= \left(\widehat{\pi}_k + \frac{\widehat{r}_k}{\rho_{k'}(k-k')}\right)\rho_{k'} \cdot (k-k') + r_k \\ &= \widehat{\pi}_k \rho_{k'} \cdot (k-k') + r_k + \widehat{r}_k = \boldsymbol{v}^{(1)} \cdot \boldsymbol{u}_{k'} \end{aligned}$$

Firstly, we note that the distributions of $\widehat{\pi}_k$ and π_k are statistically close. Secondly, we note that the inner product value $\boldsymbol{v}^{(\mathfrak{b})} \cdot \boldsymbol{u}_{k'}$ remains the same for $\mathfrak{b} \in \{0, 1\}$. Therefore, in the first step, we switch π_k to $\widehat{\pi}_k$, where the two distributions are statistically close. Then, in the second step, we utilize the function hiding property to switch the vector from $\boldsymbol{v}^{(0)}$ to $\boldsymbol{v}^{(1)}$. □

Theorem 3 *Assuming the* SXDH *assumption holds in $\mathcal{G}$ and the* IPFE *is function hiding secure, the above construction of* (1-SK, 1-CT, 1-*Slot*)-*FE for* UAWS *is adaptively simulation secure.*

Proof idea Before going for a formal proof, we discuss a high level overview of the proof. We use a three-step approach and each step consists of a group of hybrid sequence.

– In the first step, the label function $\ell_{k,\mathsf{init}}$ is reversely sampled with the value $\boldsymbol{z}[k]M_k[\boldsymbol{x}]+\beta_k$ and it is hardwired in either $\boldsymbol{u}_{k,\mathsf{init}}$ or $\boldsymbol{v}_{k,\mathsf{init}}$, whichever is computed later.
– The second step is a loop. The purpose of the loop is to change all the honest label values $\ell_{k,t,i,j,\boldsymbol{W},q}$ to simulated ones that take the form $\ell_{k,t,i,j,\boldsymbol{W},q} = \boldsymbol{s_x}[(t, i, j, \boldsymbol{W})]\boldsymbol{s}_{k,f}[q]$ where $\boldsymbol{s_x}[(t, i, j, \boldsymbol{W})]$ is hardwired in $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$ or $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$ and $\boldsymbol{s}_{k,f}[q]$ is hardwired in $\boldsymbol{v}_{k,q}$ or $\widetilde{v}_{k,q}$. The procedure depends on the order of adversary's queries.
– After all the label values $\ell_{k,t,i,j,\boldsymbol{W},q}$ are simulated, the third step uses a few more hybrids to reversely sample $\ell_{1,\mathsf{init}}$ and $\ell_{k,\mathsf{init}}|_{k>1}$ with the hardcoded values $\boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{z}+\beta_1$ and $\beta_k|_{k>1}$ respectively. We also rearrange the elements so that the distribution of the ciphertext does not change with the occurrence of the secret key whether it comes before or after the ciphertext.

Recall that the adversary is allowed to query only a single secret key either before (SK before CT) or after (CT before SK) the challenge ciphertext. Accordingly, we consider two different cases depending on the occurrence of the single secret key query.

Case 1 (CT *before* SK) : In this case, we place the reversely sampled $\ell_{k,\mathsf{init}}$ in the $\boldsymbol{v}_{k,\mathsf{init}}$ in the exponent of $\mathbb{G}_2$. The loop of the second step runs over $(k, t, i, j, \boldsymbol{W})$ in lexicographical order. In each iteration, we clean $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$ and shift everything to $\boldsymbol{v}_{k,q}$ in one step and truly randomize the label values using DDH in $\mathbb{G}_2$ and then change these to their simulated form $\ell_{k,t,i,j,\boldsymbol{W},q} = \boldsymbol{s_x}[(t, i, j, \boldsymbol{W})]\boldsymbol{s}_{k,f}[q]$ by again using DDH in $\mathbb{G}_2$. Finally, the terms $\{\boldsymbol{s_x}[(t, i, j, \boldsymbol{W})]\}_{t\in[T+1]}$ are shifted back to $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$ or $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$.

Case 2 (SK *before* CT) : In this case, we place the reversely sampled $\ell_{k,\mathsf{init}}$ in the $\boldsymbol{u}_{k,\mathsf{init}}$ in the exponent of $\mathbb{G}_1$. The second step involves a two-level loop with outer loop running over t in increasing order and inner loop running over q in increasing order. In each iteration of the loop, we move all occurrences of $\boldsymbol{r}_{k,f}[q]$ and $\boldsymbol{s}_{k,f}[q]$ into all $\boldsymbol{u}_{k,t',i',j',\boldsymbol{W}'}$ in one shot and hardwire the honest labels $\ell_{k,t,i,j,\boldsymbol{W},q}$ into $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$ for all $i, j, \boldsymbol{W}$. Next, by invoking DDH in $\mathbb{G}_1$, we first make the honest labels $\ell_{k,t,i,j,\boldsymbol{W},q}$ truly random for all $i, j, \boldsymbol{W}$ and then transform these into their simulated form $\ell_{k,t,i,j,\boldsymbol{W},q} = \boldsymbol{s_x}[(t,i,j,\boldsymbol{W})]\boldsymbol{s}_{k,f}[q]$ again by using DDH in $\mathbb{G}_1$ for all $i, j, \boldsymbol{W}$. Finally, the terms $\boldsymbol{s}_{k,f}[q]$ are shifted back to $\boldsymbol{v}_{k,q}$ or $\widetilde{\boldsymbol{v}}_{k,q}$.

We start the formal proof with the first step where both the cases can be handled together. The next two steps are managed separately according to the occurrence of the secret key. We also note that the advantage of the adversary $\mathcal{A}$ in distinguishing any two consecutive hybrids depends on either the computational hardness of the function hiding security of IPFE or the harness of the DDH assumption in source groups. Moreover, there are a few hybrids that are either identically distributed with each other or the indistinguishability follows from the security of AKGS which is an information-theoretic tool. Since there are only a polynomial number of hybrids the total advantage of the adversary in breaking the security of UAWS is bounded by a polynomial ($\mathsf{poly}(n_{\mathsf{max}}, T, N, S, 2^S, Q)$) multiplied with the advantage of an adversary in breaking function hiding security of IPFE and the hardness of the DDH assumptions in the source groups. We observe that the term 2^S remains polynomial in the security parameter for logspace Turing machines. Therefore, the security of our UAWS can be reduced to the (polynomial) security of IPFE and the hardness of the SXDH assumption.

Proof Let $\mathcal{A}$ be a PPT adversary in the security experiment of UAWS. We show that the advantage of $\mathcal{A}$ in distinguishing between the experiments $\mathsf{Expt}^{\mathsf{1\text{-}UAWS}}_{\mathcal{A},\mathsf{real}}(1^\lambda)$ and $\mathsf{Expt}^{\mathsf{1\text{-}UAWS}}_{\mathcal{A},\mathsf{ideal}}(1^\lambda)$ is negligible. In this security analysis, we additionally assume that the adversary can query only a single secret key for $(\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}})$ either before or after the challenge ciphertext. Let $((\boldsymbol{x}, 1^T, 1^{2^S}), \boldsymbol{z})$ be the challenge message and $\boldsymbol{z} \in \mathbb{Z}_p^n$. We also assume that the single key queried by the adversary cover all the indices of the ciphertexts, i.e., $\mathcal{I}_{\boldsymbol{M}} \supseteq [n]$ which is natural as the adversary gets maximum information about the ciphertext in such case. Without loss of generality and for the simplicity of exposition, we assume that the number of states in all Turing machines is the same and it is Q.

The first few hybrids are the same for both the cases: CT before SK and SK before CT. The indistinguishability arguments remain unchanged in such hybrids. In Table 3, we represent the first/last few hybrids. Let n_{max} be the maximum value of n, the length of $\boldsymbol{z}$, i.e., $\mathcal{A}$ can choose the private attribute whose maximum length can be n_{max}.

Hybrid H_0. This is the real experiment $\mathsf{Expt}^{\mathsf{1\text{-}UAWS}}_{\mathcal{A},\mathsf{real}}(1^\lambda)$ ($= \mathsf{H}_{\mathsf{real}}$ in Table 3) where the ciphertext vectors contains the challenge message $(\boldsymbol{x}, \boldsymbol{z})$ and the secret key vectors are computed using $(\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}})$.

Hybrid $\mathsf{H}_{0.1}$. At the beginning of the experiment, the challenger samples an integer $n' \leftarrow [n_{\mathsf{max}}]$ as a guess of n. This hybrid is exactly the real experiment except the challenger aborts the experiment immediately if the vector length of $\boldsymbol{z}$ is not n', i.e., $n \neq n'$. Suppose $\mathcal{A}$ outputs $\perp$ when the experiment is aborted. Thus, it is easy to see that the advantage of $\mathcal{A}$ in $\mathsf{H}_{0.1}$ is $\frac{1}{n_{\mathsf{max}}}$ times the advantage in H_0. Thus, if the advantage of $\mathcal{A}$ is negligible in H_0, then it is so in $\mathsf{H}_{0.1}$. Hence, in the remaining hybrids we simply write $n' = n$.

Hybrid $\mathsf{H}_{0.2}$. It proceeds exactly the same as $\mathsf{H}_{0.1}$ except that if the queried key $(\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}})$ is such that $(\max \mathcal{I}_{\boldsymbol{M}} > n) \wedge (\min \mathcal{I}_{\boldsymbol{M}} \leq n)$, then $\beta_k = \boldsymbol{v}_{k,\mathsf{init}}[\mathsf{acc}]$ is replaced with $\widehat{\beta_k} \leftarrow \mathbb{Z}_p$ for each $k \in \mathcal{I}_{\boldsymbol{M}}$. Thus, with high probability it holds that $\sum_{k \in \mathcal{I}_{\boldsymbol{M}}} \widehat{\beta_k} \neq 0$. The hybrids $\mathsf{H}_{0.1}$ and $\mathsf{H}_{0.2}$ are indistinguishable by the function hiding security of IPFE via the Lemma 4. Note that in this hybrid, we crucially use the randomness of the positions $\boldsymbol{v}_{k,\mathsf{init}}[\mathsf{index}_1]$ and $\boldsymbol{v}_{k,\mathsf{init}}[\mathsf{index}_2]$ (encoding the indices which are not available in the ciphertext vectors) to sample $\widehat{\beta}_k$ independently from other indices of the secret key.

Hybrid H_1. It proceeds exactly the same as $\mathsf{H}_{0.2}$ except $\ell_{k,\mathsf{init}}$ is hardwired in $\boldsymbol{v}_{k,\mathsf{init}}$ or $\boldsymbol{u}_{k,\mathsf{init}}$, and $\boldsymbol{s}_{k,f} \leftarrow \mathbb{Z}_p^Q, \boldsymbol{s}_{\boldsymbol{x}} \leftarrow \mathbb{Z}_p^{[T+1]\times[N]\times[S]\times\{0,1\}^S}$ are embedded in $\boldsymbol{v}_{k,q}, \widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$ respectively. The first change sets the stage for $\ell_{k,\mathsf{init}}$ to be reversely sampled in the next hybrid and the second change prepares the $\ell_{k,t,i,j,\boldsymbol{W},q}|_{t\leq T}, \ell_{k,T+1,i,j,\boldsymbol{W},q}$ to be simulated as pseudorandom values in the loop hybrids. More specifically, the changes are implemented as follows:

- For CT before SK, $\boldsymbol{u}_{k,\mathsf{init}}$ is set to 1 during encryption and $\boldsymbol{v}_{k,\mathsf{init}}$ is set to $\boldsymbol{r}_{\boldsymbol{x}}[(0, 1, 1, \boldsymbol{0}_S)]\boldsymbol{r}_{k,f}[1]$ during key generation.
- For SK before CT, $\boldsymbol{v}_{k,\mathsf{init}}$ is set to 1 during key generation and $\boldsymbol{u}_{k,\mathsf{init}}$ is set to $\boldsymbol{r}_{\boldsymbol{x}}[(0, 1, 1, \boldsymbol{0}_S)]\boldsymbol{r}_{k,f}[1]$ during encryption. Note that, $\boldsymbol{r}_{k,f}[1]$s are known only for $k \in \mathcal{I}_{\boldsymbol{M}}$. Thus, $\boldsymbol{u}_{k,\mathsf{init}}[\mathsf{init}]$ is unchanged in this and in the rest of the hybrids for $k \in [n] \setminus \mathcal{I}_{\boldsymbol{M}}$.
- Also, $\boldsymbol{v}_{k,q}[\mathsf{sim}]$ is set to $\boldsymbol{s}_{k,f}[q]$ and $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}[\mathsf{sim}]$ is set to $\boldsymbol{s}_{\boldsymbol{x}}[(T+1, i, j, \boldsymbol{W})]$.

Note that, the inner products between $\boldsymbol{v}$'s and $\boldsymbol{u}$'s remain unchanged. Therefore, the function hiding property of IPFE ensures that H_0 and H_1 are indistinguishable.

Hybrid H_2. It proceeds identically to H_1 except that $\ell_{k,\mathsf{init}}$ is reversely sampled from the other labels. By the piecewise security of AKGS, the hybrids H_1 and H_2 are indistinguishable (Tables 1, 2). □

Hybrid H_4. It proceeds identically to H_2 except the inner products $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}} \cdot \boldsymbol{v}_{k,q}$ and $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}} \cdot \widetilde{\boldsymbol{v}}_{k,q}$ change from the honest to simulated labels $\boldsymbol{s}_{\boldsymbol{x}}[(t, i, j, \boldsymbol{W})]\boldsymbol{s}_{k,f}[q]$ and $\boldsymbol{s}_{\boldsymbol{x}}[(T+1, i, j, \boldsymbol{W})]\boldsymbol{s}_{k,f}[q]$ respectively. This is implemented by clearing the values at $\mathsf{rand}, \mathsf{acc}, \mathsf{tb}_\tau$ of the vectors $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}, \widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$ and embedding $\boldsymbol{s}_{k,f}[q], \boldsymbol{s}_{\boldsymbol{x}}[(t, i, j, \boldsymbol{W})]$ at $\widetilde{\boldsymbol{v}}_{k,q}[\mathsf{sim}], \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}[\mathsf{sim}]$ respectively. We show the indistinguishability between the hybrids H_2 and H_3 in two separate claims:

Claim 1 *In the case of* CT *before* SK, $H_2 \approx H_4$.

Claim 2 *In the case of* SK *before* CT, $H_2 \approx H_4$.

Hybrid H_5. It proceeds exactly the same as H_4 except the values at $\mathsf{rand}, \mathsf{acc}, \mathsf{tb}_\tau$ of the vectors $\boldsymbol{v}_{k,q}, \widetilde{\boldsymbol{v}}_{k,q}$ are cleared and $\boldsymbol{u}_{k,\mathsf{init}}[\mathsf{sim}]$ is set to 1. Also, for the case of CT before SK, $\ell_{k,\mathsf{init}}$ is shifted from $\boldsymbol{v}_{k,\mathsf{init}}[\mathsf{init}]$ to $\boldsymbol{v}_{k,\mathsf{init}}[\mathsf{sim}]$. While the former change is common for both cases, the later prepares the ideal game for the case of CT before SK. Note that, the inner products between $\boldsymbol{v}$'s and $\boldsymbol{u}$'s remain unchanged. Therefore, the function hiding property of IPFE ensures that H_4 and H_5 are indistinguishable.

Hybrid H_6. It is the same as H_5 except the hardcoded values used in the reverse sampling procedure while computing $\ell_{k,\mathsf{init}}$ (for both cases). It computes $\ell_{k,\mathsf{init}}$ as follows:

$$\ell_{1,\mathsf{init}} \leftarrow \mathsf{RevSamp}((M_1, 1^N, 1^T, 1^{2^S}), \boldsymbol{x}, \boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{z} + \beta_1, (\ell_{k,t,\theta_k})_{t\in[T+1],\theta_k\in\mathcal{C}_{M_k,N,S}})$$
$$\ell_{k,\mathsf{init}} \leftarrow \mathsf{RevSamp}((M_k, 1^N, 1^T, 1^{2^S}), \boldsymbol{x}, \beta_k, (\ell_{k,t,\theta_k})_{t\in[T+1],\theta_k\in\mathcal{C}_{M_k,N,S}}), \quad \text{for all } k > 1$$

where all the other label values $\ell_{k,t,i,j,\boldsymbol{W},q} = \boldsymbol{s}_{\boldsymbol{x}}[(t, i, j, \boldsymbol{W})]\boldsymbol{s}_{k,f}[q]$ are already simulated. If the queried key satisfies the permissiveness, i.e., $\mathcal{I}_{\boldsymbol{M}} \subseteq [n]$, then this is accomplished by

Table 1 The initial few hybrids in the security proof of 1-UAWS

Hybrid	Vector	init	rand, acc, tb_τ	sim
$\mathsf{H}_{0.2}$	$\boldsymbol{v}_{k,\mathsf{init}}$	$\boldsymbol{r}_{k,f}[1]$	Normal	0
	$\boldsymbol{v}_{k,q}$		Normal	0
	$\widetilde{\boldsymbol{v}}_{k,q}$		Normal	0
	$\boldsymbol{u}_{k,\mathsf{init}}$	$\boldsymbol{r}_{\boldsymbol{x}}[(0,1,1,\boldsymbol{0}_S)]$	Normal	0
	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$		Normal	0
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		Normal	0
H_1	$\boldsymbol{v}_{k,\mathsf{init}}$	$\boxed{1}$ or $\boxed{\boldsymbol{r}_{\boldsymbol{x}}[(0,1,1,\boldsymbol{0}_S)]\boldsymbol{r}_{k,f}[1]+\beta_k(\text{or }\widehat{\beta}_k)}$	$\boxed{0,0,0}$	0
	$\boldsymbol{v}_{k,q}$		Normal	$\boxed{\boldsymbol{s}_{k,f}[q]}$
	$\widetilde{\boldsymbol{v}}_{k,q}$		Normal	0
	$\boldsymbol{u}_{k,\mathsf{init}}$	$\boxed{\boldsymbol{r}_{\boldsymbol{x}}[(0,1,1,\boldsymbol{0}_S)]\boldsymbol{r}_{k,f}[1]+\beta_k(\text{or }\widehat{\beta}_k)}$ or $\boxed{1}$	$\boxed{0,0,0}$	0
	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$		Normal	0
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		Normal	$\boxed{\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]}$
H_2	$\boldsymbol{v}_{k,\mathsf{init}}$	1 or $\boxed{\ell_{k,\mathsf{init}} \leftarrow \mathsf{RevSamp}(\boldsymbol{z}[k]M_k(\boldsymbol{x})+\beta_k(\text{or }\widehat{\beta}_k))}$	$0,0,0$	0
	$\boldsymbol{v}_{k,q}$		Normal	$\boldsymbol{s}_{k,f}[q]$
	$\widetilde{\boldsymbol{v}}_{k,q}$		Normal	0
	$\boldsymbol{u}_{k,\mathsf{init}}$	$\boxed{\ell_{k,\mathsf{init}} \leftarrow \mathsf{RevSamp}(\boldsymbol{z}[k]M_k(\boldsymbol{x})+\beta_k(\text{or }\widehat{\beta}_k))}$ or 1	$0,0,0$	0
	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$		Normal	0

Table 1 continued

Hybrid	Vector	init	rand, acc, tb_τ	sim
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		Normal	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$
Loop	$\cdots$	$\cdots$	$\cdots$	$\cdots$
H_4	$\boldsymbol{v}_{k,\mathsf{init}}$	1 or $\ell_{k,\mathsf{init}} \leftarrow \mathsf{RevSamp}(z[k]M_k(\boldsymbol{x})+\beta_k(\text{or } \widehat{\beta}_k))$	$0, 0, 0$	0
	$\boldsymbol{v}_{k,q}$		Normal	$\boldsymbol{s}_{k,f}[q]$
	$\widetilde{\boldsymbol{v}}_{k,q}$		Normal	$\boxed{\boldsymbol{s}_{k,f}[q]}$
	$\boldsymbol{u}_{k,\mathsf{init}}$	$\ell_{k,\mathsf{init}} \leftarrow \mathsf{RevSamp}(z[k]M_k(\boldsymbol{x})+\beta_k(\text{or } \widehat{\beta}_k))$ or 1	$0, 0, 0$	0
	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$		$\boxed{0, 0, 0}$	$\boxed{\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]}$
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		$\boxed{0, 0, -}$	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$
H_5	$\boldsymbol{v}_{k,\mathsf{init}}$	1 or $\boxed{0}$	$0, 0, 0$	0 or $\boxed{\ell_{k,\mathsf{init}} \leftarrow \mathsf{RevSamp}(z[k]M_k(\boldsymbol{x})+\beta_k(\text{or } \widehat{\beta}_k))}$
	$\boldsymbol{v}_{k,q}$		$\boxed{0, 0, 0}$	$\boldsymbol{s}_{k,f}[q]$
	$\widetilde{\boldsymbol{v}}_{k,q}$		$\boxed{0, 0, -}$	$\boldsymbol{s}_{k,f}[q]$

Table 1 continued

Hybrid	Vector	init	rand, acc, tb_τ	sim
	$\boldsymbol{u}_{k,\mathsf{init}}$	$\ell_{k,\mathsf{init}} \leftarrow \mathsf{RevSamp}(\boldsymbol{z}[k]M_k(\boldsymbol{x}) + \beta_k(\text{or } \widehat{\beta}_k))$ or 1	0, 0, 0	$\boxed{1}$
	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$		0, 0, 0	$\boldsymbol{s}_{\boldsymbol{x}}[(t, i, j, \boldsymbol{W})]$
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		0, 0, –	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1, i, j, \boldsymbol{W})]$
H_6	$\boldsymbol{v}_{1,\mathsf{init}}$	1 or 0	0, 0, 0	0 or $\boxed{\ell_{1,\mathsf{init}} \leftarrow \mathsf{RevSamp}(\boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{z} + \beta'_1)}$
	$\boldsymbol{v}_{k>1,\mathsf{init}}$	1 or 0	0, 0, 0	0 or $\boxed{\ell_{k,\mathsf{init}} \leftarrow \mathsf{RevSamp}(\beta'_k)}$
	$\boldsymbol{v}_{k,q}$		0, 0, 0	$\boldsymbol{s}_{k,f}[q]$
	$\widetilde{\boldsymbol{v}}_{k,q}$		0, 0, –	$\boldsymbol{s}_{k,f}[q]$
	$\boldsymbol{u}_{1,\mathsf{init}}$	$\boxed{\ell_{1,\mathsf{init}} \leftarrow \mathsf{RevSamp}(\boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{z} + \beta'_1)}$ or 1	0, 0, 0	1
	$\boldsymbol{u}_{k>1,\mathsf{init}}$	$\boxed{\ell_{k,\mathsf{init}} \leftarrow \mathsf{RevSamp}(\beta'_k)}$ or 1	0, 0, 0	1
	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$		0, 0, 0	$\boldsymbol{s}_{\boldsymbol{x}}[(t, i, j, \boldsymbol{W})]$
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		0, 0, –	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1, i, j, \boldsymbol{W})]$

Table 2 The last few hybrids in the security proof of 1-UAWS

hybrid	vector	init	rand, acc, $\mathsf{tb}_\mathcal{T}$	sim
$\mathsf{H}_{\mathsf{ideal}}$	$\boldsymbol{v}_{1,\mathsf{init}}$	0	$0,0,0$	$\ell_{1,\mathsf{init}} \leftarrow \mathsf{RevSamp}(\boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{z} + \beta'_1)$
	$\boldsymbol{v}_{k>1,\mathsf{init}}$	0	$0,0,0$	$\ell_{k,\mathsf{init}} \leftarrow \mathsf{RevSamp}(\beta'_k)$
	$\boldsymbol{v}_{k,q}$		$0,0,0$	$\boldsymbol{s}_{k,f}[q]$
	$\widetilde{\boldsymbol{v}}_{k,q}$		$0,0,-$	$\boldsymbol{s}_{k,f}[q]$
	$\boldsymbol{u}_{k,\mathsf{init}}$	$\boxed{\boldsymbol{r}_{\boldsymbol{x}}[(0,1,1,\boldsymbol{0}_S)]}$	$\boxed{\text{normal}}$	1
	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$		$\boxed{\text{normal}}$	$\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]$
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		$\boxed{\text{normal}(\boldsymbol{\nu})}$	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$
for CT before SK the sequence of hybrids ends here, i.e., $\mathsf{H}_{\mathsf{ideal}}$ is the ideal world				
H_7	$\boldsymbol{v}_{k,\mathsf{init}}$	1	$0,0,0$	0
	$\boldsymbol{v}_{k,q}$		$0,0,0$	$\boldsymbol{s}_{k,f}[q]$
	$\widetilde{\boldsymbol{v}}_{k,q}$		$0,0,-$	$\boldsymbol{s}_{k,f}[q]$
	$\boldsymbol{u}_{1,\mathsf{init}}$	$\boxed{\ell_{1,\mathsf{init}} \leftarrow \mathsf{RevSamp}(\boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{d} + \beta'_1)}$	$0,0,0$	1
	$\boldsymbol{u}_{k>1,\mathsf{init}}$	$\ell_{k,\mathsf{init}} \leftarrow \mathsf{RevSamp}(\beta'_k)$	$0,0,0$	1
	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$		$0,0,0$	$\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]$
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		$0,0,-$	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$
for SK before CT traverse in reverse direction until H_0 with $\boldsymbol{d}$ in place of $\boldsymbol{z}$				
$\mathsf{H}_{(7\to 0)}$	$\boldsymbol{v}_{k,\mathsf{init}}$	$\boxed{\boldsymbol{r}_{k,f}[1]}$	$\boxed{\text{normal}}$	0
	$\boldsymbol{v}_{k,q}$		$\boxed{\text{normal}}$	$\boxed{0}$
	$\widetilde{\boldsymbol{v}}_{k,q}$		$\boxed{\text{normal}}$	$\boxed{0}$
	$\boldsymbol{u}_{k,\mathsf{init}}$	$\boxed{\boldsymbol{r}_{\boldsymbol{x}}[(0,1,1,\boldsymbol{0}_S)]}$	$\boxed{\text{normal}}$	$\boxed{0}$
	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$		$\boxed{\text{normal}}$	$\boxed{0}$
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		$\boxed{\text{normal}(\boldsymbol{d})}$	$\boxed{0}$
$\mathsf{H}_{\mathsf{ideal}}$	$\boldsymbol{v}_{k,\mathsf{init}}$	$\boldsymbol{r}_{k,f}[1]$	normal	0
	$\boldsymbol{v}_{k,q}$		normal	0
	$\widetilde{\boldsymbol{v}}_{k,q}$		normal	0
	$\boldsymbol{u}_{k,\mathsf{init}}$	$\boldsymbol{r}_{\boldsymbol{x}}[(0,1,1,\boldsymbol{0}_S)]$	normal	$\boxed{1}$
	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$		normal	$\boxed{\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]}$
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		$\text{normal}(\boldsymbol{d})$	$\boxed{\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]}$
for SK before CT, $\mathsf{H}_{\mathsf{ideal}}$ is the ideal world				

The "normal" in	rand	acc	$\mathsf{tb}_\mathcal{T}$
$\boldsymbol{v}_{k,\mathsf{init}}$:	0	β_k	0
$\boldsymbol{v}_{k,q}$:	$-\boldsymbol{r}_{k,f}[q]$	0	$(\mathbf{M}_{k,\mathcal{T}}\boldsymbol{r}_{k,f})[q]$
$\widetilde{\boldsymbol{v}}_{k,q}$:	$-\boldsymbol{r}_{k,f}[q]$	$\boldsymbol{y}_k[q]$	$-$
$\boldsymbol{u}_{k,\mathsf{init}}$:	0	1	0
$t \le T,\ \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$:	$\boldsymbol{r}_{\boldsymbol{x}}[(t-1,i,j,\boldsymbol{W})]$	0	$c_\mathcal{T}(\boldsymbol{x},t,i,j,\boldsymbol{W};\boldsymbol{r}_{\boldsymbol{x}})$
if normal, $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$:	$\boldsymbol{r}_{\boldsymbol{x}}[(T,i,j,\boldsymbol{W})]$	$\boldsymbol{z}[k]$	$-$
if normal($\boldsymbol{\omega}$), $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$:	$\boldsymbol{r}_{\boldsymbol{x}}[(T,i,j,\boldsymbol{W})]$	$\boldsymbol{\omega}[k]$	$-$

a statistical transformation on $\{\beta_k : \beta_k \leftarrow \mathbb{Z}_p, \sum_{k\in\mathcal{I}_{\boldsymbol{M}}} \beta_k = 0\}$. We replace β_k by newly sampled β_k:

$$\beta_1 = \beta'_1 - z[1]M_1(\boldsymbol{x}) + \boldsymbol{M}(\boldsymbol{x})^\top z$$
$$\beta_k = \beta'_k - z[k]M_k(\boldsymbol{x}) \quad \text{for all } k > 1$$

Table 3 The remaining note of the first/last few hybrids in the security proof of 1-UAWS.

In H_1,	$\boxed{\mathsf{SK\ before\ CT}}$	$\boxed{\mathsf{CT\ before\ SK}}$	In $\mathsf{H}_2, \mathsf{H}_4$,	$\boxed{\mathsf{SK\ before\ CT}}$	$\boxed{\mathsf{CT\ before\ SK}}$
$\boldsymbol{v}_{k,\mathsf{init}}[\mathsf{init}] =$	1	$\boldsymbol{r}_{\boldsymbol{x}}[(0,1,1,\boldsymbol{0}_S)]\boldsymbol{r}_{k,f}[1]$	$\boldsymbol{v}_{k,\mathsf{init}}[\mathsf{init}] =$	1	$\mathsf{RevSamp}(\alpha)$
$\boldsymbol{u}_{k,\mathsf{init}}[\mathsf{init}] =$	$\boldsymbol{r}_{\boldsymbol{x}}[(0,1,1,\boldsymbol{0}_S)]\boldsymbol{r}_{k,f}[1]$	1	$\boldsymbol{u}_{k,\mathsf{init}}[\mathsf{init}] =$	$\mathsf{RevSamp}(\alpha)$	1
In H_5,	$\boxed{\mathsf{SK\ before\ CT}}$	$\boxed{\mathsf{CT\ before\ SK}}$	In H_5,	$\boxed{\mathsf{SK\ before\ CT}}$	$\boxed{\mathsf{CT\ before\ SK}}$
$\boldsymbol{v}_{k,\mathsf{init}}[\mathsf{init}] =$	1	0	$\boldsymbol{v}_{k,\mathsf{init}}[\mathsf{sim}] =$	0	$\mathsf{RevSamp}(\alpha)$
$\boldsymbol{u}_{k,\mathsf{init}}[\mathsf{init}] =$	$\mathsf{RevSamp}(\alpha)$	1	$\boldsymbol{u}_{k,\mathsf{init}}[\mathsf{sim}] =$	1	1

The "$\mathsf{RevSamp}(\alpha)$" means: $\ell_{k,\mathsf{init}} \leftarrow \mathsf{RevSamp}((M_k, 1^N, 1^T, 1^{2^S}), \boldsymbol{x}, \alpha, (\ell_{k,t,\theta_k})_{t\in[T+1],\theta_k\in\mathcal{C}_{M_k,N,S}})$

In H_2, $\mathsf{H}_{\mathsf{ideal}}$, $\ell_{k,t,i,j,\boldsymbol{W},q} = L_{k,t,i,j,\boldsymbol{W},q}(\boldsymbol{x})$ are computed honestly using IPFE.

In $\mathsf{H}_4, \mathsf{H}_5, \mathsf{H}_6$, $\mathsf{H}_{\mathsf{ideal}}$, $\ell_{k,t,i,j,\boldsymbol{W},q} = \boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]\boldsymbol{s}_{k,f}[q]$ are simulated and computed using IPFE.

In $\mathsf{H}_{\mathsf{ideal}}$, the positions $\mathsf{rand}, \mathsf{acc}, \mathsf{tb}_\tau$ of $\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}, \widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$ are kept either normal or normal$(\boldsymbol{\nu})$, for an arbitrary vector $\boldsymbol{\nu}$, due to security proof of $\mathsf{PK\text{-}UAWS}$. These entries have no effect in this simulation.

In $\mathsf{H}_{\mathsf{ideal}}$, the position sim of $\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}, \widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$ are kept $1, \boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})], \boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$ respectively due to security proof of $\mathsf{PK\text{-}UAWS}$. These entries have no effect in this simulation.

where $\beta'_k \leftarrow \mathbb{Z}_p$. Observe that it still holds that $\sum_{k\in\mathcal{I}_{\boldsymbol{M}}} \beta_k = 0$. On the other hand, if the key under consideration does not satisfy the permissiveness, i.e., $(\max \mathcal{I}_{\boldsymbol{M}} > n) \wedge (\min \mathcal{I}_{\boldsymbol{M}} < n)$, then we know that $\widehat{\beta}_k$ are uniform over $\mathbb{Z}_p$. Thus, we can replace $\widehat{\beta}_k$ by new $\widehat{\beta}_k$:

$$\widehat{\beta}_1 = \beta'_1 - z[1]M_1(\boldsymbol{x}) + \boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{z}$$
$$\widehat{\beta}_k = \beta'_k - z[k]M_k(\boldsymbol{x}) \quad \text{for all } k > 1$$

where $\beta'_k \leftarrow \mathbb{Z}_p$. Note that, the distributions of new β_k or $\widehat{\beta}_k$ are statistically close to their old versions and hence the two hybrids H_5 and H_6 are indistinguishable.

Hybrid $\mathsf{H}_{\mathsf{ideal}}$. This hybrid is equivalent to the ideal experiment $\mathsf{Expt}^{\mathsf{1\text{-}UAWS}}_{\mathcal{A},\mathsf{ideal}}(1^\lambda)$ for the case of CT before SK. Thus, one should omit this hybrid in the case of SK before CT. In $\mathsf{H}_{\mathsf{ideal}}$, the positions $\mathsf{init}, \mathsf{rand}, \mathsf{acc}, \mathsf{tb}_\tau$ of the vectors $\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}, \widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$ are changed back to their normal form as they were in H_0 except we use an arbitrary vector $\boldsymbol{\nu} \leftarrow \mathbb{Z}_p^n$ in place of $\boldsymbol{z}$ (for $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$). This change has no effect in the inner products between $\boldsymbol{u}$'s and $\boldsymbol{v}$'s since the corresponding terms in $\boldsymbol{v}$'s are zero. The purpose of this change is to maintain the distribution of the ciphertext vectors consistent with the case of SK before CT. Finally, $\mathsf{H}_{\mathsf{ideal}}$ is indistinguishable from H_6 by the function hiding property of IPFE, and hence $\mathsf{H}_0 = \mathsf{Expt}^{\mathsf{1\text{-}UAWS}}_{\mathcal{A},\mathsf{real}}(1^\lambda) \approx \mathsf{H}_{\mathsf{ideal}}$.

The sequence of hybrids for the case of CT before SK ends here and the rest of the hybrids are required only to handle the case of SK before CT.

Hybrid H_7. It proceeds exactly the same as H_6 except it samples a dummy vector $\boldsymbol{d} \leftarrow \mathbb{Z}_p^n$ such that

$$\boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{z} = \boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{d} = \sum_{k\in[n]} M_k(\boldsymbol{x})\boldsymbol{d}[k].$$

and reversely sample $\ell_{1,\mathsf{init}}$ with the hardcoded value $\boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{d} + \beta_1$ instead of $\boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{z} + \beta_1$. Note that, this is statistical change to the computation of $\ell_{1,\mathsf{init}}$, and hence the hybrids H_6 and H_7 are indistinguishable to the adversary.

Hybrid $\mathsf{H}_{(7\to 0)}$. Next, for the case of SK before CT, we traverse in the reverse direction from H_7 to all the way to H_0 with the dummy vector $\boldsymbol{d}$ in place of $\boldsymbol{z}$. This step is inspired from the proof technique used by Datta and Pal [38]. We skip the descriptions of these hybrids as the indistinguishability arguments would be exactly similar to what we used for reaching H_7 from

H_0. We denote the *new* H_0 as $\mathsf{H}_{(7\to 0)}$ and the hybrids H_7 and $\mathsf{H}_{(7\to 0)}$ are indistinguishable by the function hiding security of IPFE and the piecewise security of AKGS. After this hybrid, observe that the reduction do not need to guess n which enables the final simulator to generate the pre-ciphertext secret key without any information about the length of private attribute z.

Hybrid $\mathsf{H}_{\mathsf{ideal}}$. It is exactly the same as $\mathsf{H}_{(7\to 0)}$ except the position sim of the vectors $\boldsymbol{u}_{k,\mathsf{init}}$, $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$ and $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$ are set as 1, $\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]$ and $\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$ respectively. Observe that this change has no effect in the inner product computation of these vectors with their corresponding vectors in the secret key as the positions in the secret key vectors are zero. This, however, keeps the ciphertext distribution consistent with the case of CT before SK. Therefore, $\mathsf{H}_{\mathsf{ideal}}$ and $\mathsf{H}_{(7\to 0)}$ are indistinguishable by the function hiding security of the IPFE. We also note that $\mathsf{H}_{\mathsf{ideal}}$ is the ideal experiment $\mathsf{Expt}^{1\text{-}\mathsf{UAWS}}_{\mathcal{A},\mathsf{ideal}}(1^\lambda)$ for the case of SK before CT, and hence $\mathsf{H}_0 = \mathsf{Expt}^{1\text{-}\mathsf{UAWS}}_{\mathcal{A},\mathsf{real}}(1^\lambda) \approx \mathsf{H}_{\mathsf{ideal}}$. This completes the proof. □ □

Proof of Claim 1 For the case of CT before SK, we prove $\mathsf{H}_2 \approx \mathsf{H}_4$ using a sequence of hybrids $\mathsf{H}_{3,t,i,j,\boldsymbol{W},1},\ldots,\mathsf{H}_{3,t,i,j,\boldsymbol{W},5}$ for $(t,i,j,\boldsymbol{W}) \in [T]\times[N]\times[S]\times\{0,1\}^S$ in lexicographical order. These hybrids are described in Table 4. Then, we use another sequence of hybrids (dedicated for the second IPFE) $\widetilde{\mathsf{H}}_3, \widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},1},\ldots,\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},5}$ for $(t,i,j,\boldsymbol{W}) \in [T]\times[N]\times[S]\times\{0,1\}^S$ in lexicographical order. These hybrids are illustrated in Table 5. We denote by $(t,i,j,\boldsymbol{W})+1$ the next tuple of indices in increasing order. We observe that $\boldsymbol{u}$'s are listed before $\boldsymbol{v}$'s since in the case of CT before SK the ciphertext appears before the secret key.

Hybrid $\mathsf{H}_{3,t,i,j,\boldsymbol{W},1}$. It proceeds identically to H_2 except that for all $(t',i',j',\boldsymbol{W}') < (t,i,j,\boldsymbol{W})$, $\boldsymbol{u}_{k,t',i',j',\boldsymbol{W}'}$ has its values in rand and tb_τ's cleared, and that a random value $\boldsymbol{s}_{\boldsymbol{x}}[(t',i',j',\boldsymbol{W}')]$ is embedded in $\boldsymbol{u}_{k,t',i',j',\boldsymbol{W}'}[\mathsf{sim}]$. This means that all the labels for $(t',i',j',\boldsymbol{W}') < (t,i,j,\boldsymbol{W})$ are simulated, the first label $\ell_{k,\mathsf{init}}$ is reversely sampled and the rest are honestly computed.

Hybrid $\mathsf{H}_{3,t,i,j,\boldsymbol{W},2}$. It proceeds exactly the same way as $\mathsf{H}_{3,t,i,j,\boldsymbol{W},1}$ except that the values in $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$ are set to zero and its inner product with $\boldsymbol{v}_{k,q}$'s, i.e. the labels $\ell_{k,t,i,j,\boldsymbol{W},q}$ for all k,q, are hardcoded into $\boldsymbol{v}_{k,q}$'s as follows:

- The positions rand and tb_τ of $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$ are set to 0.
- The value at $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}[\mathsf{sim}^{\mathsf{temp}}]$ is set to 1.
- The honest labels $\ell_{k,t,i,j,\boldsymbol{W},q} = -\boldsymbol{r}_{\boldsymbol{x}}[(t-1,i,j,\boldsymbol{W})]\boldsymbol{r}_{k,f}[q] + \cdots$ are embedded in $\boldsymbol{v}_{k,q}[\mathsf{sim}^{\mathsf{temp}}]$ for each $q \in [Q]$ and $k \in \mathcal{I}_{\boldsymbol{M}}$ where "$\cdots$" represents $\sum_{\tau\in\mathcal{T}} c_\tau(\boldsymbol{x};t,i,j,\boldsymbol{W};\boldsymbol{r}_{\boldsymbol{x}})(\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q]$.

As one can verify that the inner products between the vectors are unchanged, the indistinguishability between the hybrids $\mathsf{H}_{3,t,i,j,\boldsymbol{W},1}$ and $\mathsf{H}_{3,t,i,j,\boldsymbol{W},2}$ is guaranteed by the function hiding security of IPFE.

Hybrid $\mathsf{H}_{3,t,i,j,\boldsymbol{W},3}$. It proceeds similar to $\mathsf{H}_{3,t,i,j,\boldsymbol{W},2}$ except that the labels $\ell_{k,t,i,j,\boldsymbol{W},q}$ are changed to truly randomized values. We can invoke DDH assumption in $\mathbb{G}_2$ between the hybrids since the random values $\boldsymbol{r}_{\boldsymbol{x}}[(t-1,i,j,\boldsymbol{W})]$ and $\boldsymbol{r}_{k,f}[q]$'s only appear in the exponent of $\mathbb{G}_2$: for each $k \in \mathcal{I}_{\boldsymbol{M}}$, given an $\mathsf{MDDH}_{1,q}$ challenge

$$[\![\boldsymbol{r}_{k,f}[1],\ldots,\boldsymbol{r}_{k,f}[Q];\ \Delta_{k,1},\ldots,\Delta_{k,Q}]\!]_2 :$$
$$\Delta_{k,q}\begin{cases} = \boldsymbol{r}_{\boldsymbol{x}}[(t-1,i,j,\boldsymbol{W})]\boldsymbol{r}_{k,f}[q], & \text{if DDH tuple}\\ \leftarrow \mathbb{Z}_p, & \text{if truly random tuple}\end{cases}$$

Table 4 The loop hybrids for $t \leq T$ in the security proof of 1-UAWS for the case where the ciphertext challenge comes before the secret key query

hybrid	vector	rand	tb_τ	sim	$\mathsf{sim}^{\mathsf{temp}}$
$\mathsf{H}_{3,t,i,j,\boldsymbol{W},1}$	$\boldsymbol{u}_{k,t',i',j',\boldsymbol{W}'} < (t,i,j,\boldsymbol{W})$	0	0	$\boldsymbol{s_x}[(t',i',j',\boldsymbol{W}')]$	0
	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$	$\boldsymbol{r_x}[(t-1,i,j,\boldsymbol{W})]$	$c_\tau(\boldsymbol{x},t,i,j,\boldsymbol{W};\boldsymbol{r_x})$	0	0
	$\boldsymbol{u}_{k,t',i',j',\boldsymbol{W}'} > (t,i,j,\boldsymbol{W})$	$\boldsymbol{r_x}[(t'-1,i',j',\boldsymbol{W}')]$	$c_\tau(\boldsymbol{x},t',i',j',\boldsymbol{W}';\boldsymbol{r_x})$	0	0
	$\boldsymbol{v}_{k,q}$	$-\boldsymbol{r}_{k,f}[q]$	$(\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q]$	$\boldsymbol{s}_{k,f}[q]$	0
$\mathsf{H}_{3,t,i,j,\boldsymbol{W},2}$	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$	$\boxed{0}$	$\boxed{0}$	0	$\boxed{1}$
	$\boldsymbol{v}_{k,q}$	$-\boldsymbol{r}_{k,f}[q]$	$(\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q]$	$\boldsymbol{s}_{k,f}[q]$	$\boxed{\text{honest } \ell_{k,t,i,j,\boldsymbol{W},q} = -\boldsymbol{r_x}[(t-1,i,j,\boldsymbol{W})]\boldsymbol{r}_{k,f}[q] + \cdots}$
$\mathsf{H}_{3,t,i,j,\boldsymbol{W},3}$	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$	0	0	0	1
	$\boldsymbol{v}_{k,q}$	$-\boldsymbol{r}_{k,f}[q]$	$(\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q]$	$\boldsymbol{s}_{k,f}[q]$	$\boxed{\ell_{k,t,i,j,\boldsymbol{W},q} \xleftarrow{\$} \mathbb{Z}_p}$
$\mathsf{H}_{3,t,i,j,\boldsymbol{W},4}$	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$	0	0	0	1
	$\boldsymbol{v}_{k,q}$	$-\boldsymbol{r}_{k,f}[q]$	$(\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q]$	$\boldsymbol{s}_{k,f}[q]$	$\boxed{\text{simulated } \ell_{k,t,i,j,\boldsymbol{W},q} = \boldsymbol{s_x}[(t,i,j,\boldsymbol{W})]\boldsymbol{s}_{k,f}[q]}$
$\mathsf{H}_{3,t,i,j,\boldsymbol{W},5} \equiv \mathsf{H}_{3,t',i',j',\boldsymbol{W}',1}$ for $(t',i',j',\boldsymbol{W}') = (t,i,j,\boldsymbol{W})+1$	$\boldsymbol{u}_{k,t',i',j',\boldsymbol{W}'} < (t,i,j,\boldsymbol{W})$	0	0	$\boldsymbol{s_x}[(t',i',j',\boldsymbol{W}')]$	0
	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$	0	0	$\boxed{\boldsymbol{s_x}[(t,i,j,\boldsymbol{W})]}$	$\boxed{0}$
	$\boldsymbol{u}_{k,t',i',j',\boldsymbol{W}'} > (t,i,j,\boldsymbol{W})$	$\boldsymbol{r_x}[(t'-1,i',j',\boldsymbol{W}')]$	$c_\tau(\boldsymbol{x},t',i',j',\boldsymbol{W}';\boldsymbol{r_x})$	0	0
	$\boldsymbol{v}_{k,q}$	$-\boldsymbol{r}_{k,f}[q]$	$(\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q]$	$\boldsymbol{s}_{k,f}[q]$	$\boxed{0}$

For brevity, $\boldsymbol{u}_{\mathsf{init}}, \boldsymbol{v}_{\mathsf{init}}, \widetilde{\boldsymbol{v}}_{k,q}, \widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}, \boldsymbol{v}_{k,q}[\mathsf{acc}] = 0, \boldsymbol{u}_{k,t\leq T,i,j,\boldsymbol{W}}[\mathsf{acc}] = 0$ are suppressed.
The reversely sampled $\ell_{k,\mathsf{init}}$ is hardwired in $\boldsymbol{u}_{k,\mathsf{init}}$, and is only needed (and can only be computed so by the reduction) in the exponent of G_2:

$[\![\ell_{k,\mathsf{init}}]\!]_2 \leftarrow \mathsf{RevSamp}((M_k, 1^N, 1^T, 1^{2^S}), \boldsymbol{x}, [\![\boldsymbol{z}[k]M_k(\boldsymbol{x}) + \beta_k]\!]_2, ([\![\ell_{k,t,\theta_k}]\!]_2)_{t\in[T+1],\theta_k\in\mathcal{C}_{M_k,N,S}})$.

In the intermediate hybrids, $\boldsymbol{u}_{k,t',i',j',\boldsymbol{W}'}$'s are suppressed. They remain unchanged in this iteration.
The omitted term "$\cdots$" at $\boldsymbol{v}_{k,q}[\mathsf{sim}^{\mathsf{temp}}]$ is $\sum_{\tau\in\mathcal{T}} c_\tau(\boldsymbol{x};t,i,j,\boldsymbol{W};\boldsymbol{r_x})(\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q]$.
In this iteration, the labels $\ell_{k,t',i',j',\boldsymbol{W}',q}$ with $(t',i',j',\boldsymbol{W}')$ are computed as:

$(t',i',j',\boldsymbol{W}') < (t,i,j,\boldsymbol{W})$:	simulated as $\boldsymbol{s_x}[(t',i',j',\boldsymbol{W}')]\boldsymbol{s}_{k,f}[q]$ and computed using IPFE
$(t',i',j',\boldsymbol{W}') = (t,i,j,\boldsymbol{W})$:	computed honestly using IPFE in $\mathsf{H}_{3,k,t,i,j,\boldsymbol{W},1}$
	computed honestly and hardwired in SK in $\mathsf{H}_{3,k,t,i,j,\boldsymbol{W},2}$
	simulated as random and hardwired in SK in $\mathsf{H}_{3,k,t,i,j,\boldsymbol{W},3}$
	simulated as $\boldsymbol{s_x}[(t,i,j,\boldsymbol{W})]\boldsymbol{s}_{k,f}[q]$ and hardwired in SK in $\mathsf{H}_{3,k,t,i,j,\boldsymbol{W},4}$
	simulated as $\boldsymbol{s_x}[(t,i,j,\boldsymbol{W})]\boldsymbol{s}_{k,f}[q]$ and computed using IPFE in $\mathsf{H}_{3,k,t,i,j,\boldsymbol{W},5}$
$(t',i',j',\boldsymbol{W}') > (t,i,j,\boldsymbol{W})$:	computed honestly using IPFE

The net effect is that $\ell_{k,t\leq T,i,j,\boldsymbol{W},q}$'s change from *honest* to *simulated.*
Note that, in this iteration, $\ell_{k,T+1,i,j,\boldsymbol{W},q}$'s are honestly computed for all $(k, T+1, i, j, \boldsymbol{W}, q)$.

we compute the labels as $\ell_{k,t,i,j,\boldsymbol{W},q} = -\Delta_{k,q} + \cdots$. If a DDH tuple is received, the labels use pseudorandom *randomizers* $\boldsymbol{r}_{t-1}[(i, j, \boldsymbol{W}, \sqcup)] = \boldsymbol{r_x}[(t-1, i, j, \boldsymbol{W})]\boldsymbol{r}_{k,f}[q]$ as in $\mathsf{H}_{3,t,i,j,\boldsymbol{W},2}$. If a truly random tuple is received, these label values are truly random *randomizers* $\boldsymbol{r}_{t-1}[(i, j, \boldsymbol{W}, \sqcup)] \leftarrow \mathbb{Z}_p^Q$ as in $\mathsf{H}_{3,t,i,j,\boldsymbol{W},3}$ due to the special piecewise security of AKGS. Note that, the values $[\![\ell_{k,\mathsf{init}}]\!]_2 \leftarrow \mathsf{RevSamp}(\cdots)$ can be efficiently computed in the exponent of $\mathbb{G}_2$.

Hybrid $\mathsf{H}_{3,t,i,j,\boldsymbol{W},4}$. It proceeds identical to $\mathsf{H}_{3,t,i,j,\boldsymbol{W},3}$ except the truly random labels $\ell_{k,t,i,j,\boldsymbol{W},q}$ for all $q \in [Q]$, $k \in \mathcal{I}_{\boldsymbol{M}}$ are replaced by pseudorandom values $\boldsymbol{s_x}[(t, i, j, \boldsymbol{W})]\boldsymbol{s}_{k,f}[q]$ with $\boldsymbol{s_x}[(t, i, j, \boldsymbol{W})] \leftarrow \mathbb{Z}_p$. The hybrids $\mathsf{H}_{3,t,i,j,\boldsymbol{W},3}$ and $\mathsf{H}_{3,t,i,j,\boldsymbol{W},4}$ are indistinguishable due to the DDH assumption in $\mathbb{G}_2$ (the argument is similar to that of in the previous hybrid).

Hybrid $\mathsf{H}_{3,t,i,j,\boldsymbol{W},5}$. It proceeds exactly the same way as $\mathsf{H}_{3,t,i,j,\boldsymbol{W},4}$ except the pseudorandom labels $\ell_{k,t,i,j,\boldsymbol{W},q} = \boldsymbol{s_x}[(t, i, j, \boldsymbol{W})]\boldsymbol{s}_{k,f}[q]$ hardwired in $\boldsymbol{v}_{k,q}[\mathsf{sim}^{\mathsf{temp}}]$'s are *split* into $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}[\mathsf{sim}]$ (embedding the factor $\boldsymbol{s_x}[(t, i, j, \boldsymbol{W})]$) and $\boldsymbol{v}_{k,q}[\mathsf{sim}]$'s (embedding the factor $\boldsymbol{s}_{k,f}[q]$). The inner products in the hybrids $\mathsf{H}_{3,t,i,j,\boldsymbol{W},4}$ and $\mathsf{H}_{3,t,i,j,\boldsymbol{W},5}$ are unchanged and hence the these two hybrids are indistinguishable due to the function hid-

Table 5 The hybrid $\widetilde{\mathsf{H}}_3$ followed by the loop hybrids in the security proof of 1-UAWS for the case where the ciphertext challenge comes before the secret key query

hybrid	vector	rand	acc	sim	sim$^{\mathsf{temp}}$
$\widetilde{\mathsf{H}}_3$	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	$\boldsymbol{r_x}[(T,i,j,\boldsymbol{W})]$	$\boldsymbol{z}[k]$	$\boxed{0}$	0
	$\widetilde{\boldsymbol{v}}_{k,q}$	$-\boldsymbol{r}_{k,f}[q]$	$\boldsymbol{y}_k[q]$	$\boxed{\boldsymbol{s}_{k,f}[q]}$	0
$\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},1}$	$\widetilde{\boldsymbol{u}}_{k,\,T\,+\,1,\,i',\,j',\,\boldsymbol{W}' < (i,\,j,\,\boldsymbol{W})}$	0	0	$\boldsymbol{s_x}[(T+1,i',j',\boldsymbol{W}')]$	0
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	$\boldsymbol{r_x}[(T,i,j,\boldsymbol{W})]$	$\boldsymbol{z}[k]$	0	0
	$\widetilde{\boldsymbol{u}}_{k,\,T\,+\,1,\,i',\,j',\,\boldsymbol{W}' > (i,\,j,\,\boldsymbol{W})}$	$\boldsymbol{r_x}[(T,i',j',\boldsymbol{W}')]$	$\boldsymbol{z}[k]$	0	0
	$\widetilde{\boldsymbol{v}}_{k,q}$	$-\boldsymbol{r}_{k,f}[q]$	$\boldsymbol{y}_k[q]$	$\boldsymbol{s}_{k,f}[q]$	0
$\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},2}$	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	$\boxed{0}$	$\boxed{0}$	0	$\boxed{1}$
	$\widetilde{\boldsymbol{v}}_{k,q}$	$-\boldsymbol{r}_{k,f}[q]$	$\boldsymbol{y}_k[q]$	$\boldsymbol{s}_{k,f}[q]$	$\boxed{\text{honest } \ell_{k,T+1,i,j,\boldsymbol{W},q} = -\boldsymbol{r_x}[(T,i,j,\boldsymbol{W})]\boldsymbol{r}_{k,f}[q]+\cdots}$
$\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},3}$	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	0	0	0	1
	$\widetilde{\boldsymbol{v}}_{k,q}$	$-\boldsymbol{r}_{k,f}[q]$	$\boldsymbol{y}_k[q]$	$\boldsymbol{s}_{k,f}[q]$	$\boxed{\ell_{k,T+1,i,j,\boldsymbol{W},q} \xleftarrow{\$} \mathbb{Z}_p}$
$\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},4}$	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	0	0	0	1
	$\widetilde{\boldsymbol{v}}_{k,q}$	$-\boldsymbol{r}_{k,f}[q]$	$\boldsymbol{y}_k[q]$	$\boldsymbol{s}_{k,f}[q]$	$\boxed{\text{simulated } \ell_{k,t,i,j,\boldsymbol{W},q} = \boldsymbol{s_x}[(T+1,i,j,\boldsymbol{W})]\boldsymbol{s}_{k,f}[q]}$
$\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},5}$ $\equiv$	$\widetilde{\boldsymbol{u}}_{k,\,T\,+\,1,\,i',\,j',\,\boldsymbol{W}' < (i,\,j,\,\boldsymbol{W})}$	0	0	$\boldsymbol{s_x}[(T+1,i',j',\boldsymbol{W}')]$	0
$\widetilde{\mathsf{H}}_{3,i',j',\boldsymbol{W}',1}$ for	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	0	0	$\boxed{\boldsymbol{s_x}[(T+1,i,j,\boldsymbol{W})]}$	$\boxed{0}$
$(i',j',\boldsymbol{W}')$ $=$	$\widetilde{\boldsymbol{u}}_{k,\,T\,+\,1,\,i',\,j',\,\boldsymbol{W}' > (i,\,j,\,\boldsymbol{W})}$	$\boldsymbol{r_x}[(T,i',j',\boldsymbol{W}')]$	$\boldsymbol{z}[k]$	0	0
$(i,j,\boldsymbol{W})+1$	$\widetilde{\boldsymbol{v}}_{k,q}$	$-\boldsymbol{r}_{k,f}[q]$	$\boldsymbol{y}_k[q]$	$\boldsymbol{s}_{k,f}[q]$	$\boxed{0}$

For brevity, $\boldsymbol{u}_{\mathsf{init}}, \boldsymbol{v}_{\mathsf{init}}, \boldsymbol{v}_{k,q}, \boldsymbol{u}_{k,T+1,i,j,\boldsymbol{W}}$ are suppressed.

The reversely sampled $\ell_{k,\mathsf{init}}$ is hardwired in $\boldsymbol{u}_{k,\mathsf{init}}$, and is only needed (and can only be computed so by the reduction) in the exponent of G_2:

$$[\![\ell_{k,\mathsf{init}}]\!]_2 \leftarrow \mathsf{RevSamp}((M_k, 1^N, 1^T, 1^{2^S}), \boldsymbol{x}, [\![\boldsymbol{z}[k]M_k(\boldsymbol{x})+\beta_k]\!]_2, ([\![\ell_{k,t,\theta_k}]\!]_2)_{t\in[T+1],\theta_k\in\mathcal{C}_{M_k,N,S}}).$$

In the intermediate hybrids, $\widetilde{\boldsymbol{u}}_{k,T+1,i',j',\boldsymbol{W}'}$'s are suppressed. They remain unchanged in this iteration.

The omitted term "$\cdots$" at $\widetilde{\boldsymbol{v}}_{k,q}[\mathsf{sim}^{\mathsf{temp}}]$ is $\boldsymbol{y}_k[q]\boldsymbol{z}[k]$.

In this iteration, the labels $\ell_{k,T+1,i',j',\boldsymbol{W}',q}$ with $(i',j',\boldsymbol{W}')$ are computed as:

$(i',j',\boldsymbol{W}') < (i,j,\boldsymbol{W})$: simulated as $\boldsymbol{s_x}[(T+1,i',j',\boldsymbol{W}')]\boldsymbol{s}_{k,f}[q]$ and computed using IPFE
$(i',j',\boldsymbol{W}') = (i,j,\boldsymbol{W})$: computed honestly using IPFE in $\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},1}$
computed honestly and hardwired in SK in $\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},2}$
simulated as random and hardwired in SK in $\mathsf{H}_{3,i,j,\boldsymbol{W},3}$
simulated as $\boldsymbol{s_x}[(T+1,i,j,\boldsymbol{W})]\boldsymbol{s}_{k,f}[q]$ and hardwired in the SK in $\mathsf{H}_{3,i,j,\boldsymbol{W},4}$
simulated as $\boldsymbol{s_x}[(T+1,i,j,\boldsymbol{W})]\boldsymbol{s}_{k,f}[q]$ and computed using IPFE in $\mathsf{H}_{3,i,j,\boldsymbol{W},5}$
$(i',j',\boldsymbol{W}') > (i,j,\boldsymbol{W})$: computed honestly using IPFE

The net effect is that $\ell_{k,T+1,i,j,\boldsymbol{W},q}$'s change from *honest* to *simulated*.

Note that, in this iteration, $\ell_{k,t\le T,i,j,\boldsymbol{W},q}$'s are unchanged for all $(k,t,i,j,\boldsymbol{W},q)$ and are already simulated.

The hybrid $\widetilde{\mathsf{H}}_3$ starts after the loop of Table 4 finishes, i.e. after the hybrid $\mathsf{H}_{3,T,N,S,\boldsymbol{1}_S,5}$ and the hybrid $\widetilde{\mathsf{H}}_{3,N,S,\boldsymbol{1}_S,5}$ is identical to the hybrid H_4 (c.f. Table 3).

ing security of IPFE. Moreover, it can be observed that $\mathsf{H}_{3,t,i,j,\boldsymbol{W},5} \equiv \mathsf{H}_{3,t',i',j',\boldsymbol{W}',3}$ for $(t',i',j',\boldsymbol{W}') = (t,i,j,\boldsymbol{W})+1$.

Therefore, in this sequence of hybrids for $t \le T$, we have $\mathsf{H}_{3,1,1,1,\boldsymbol{0}_S,1} \approx \mathsf{H}_{3,T,N,S,\boldsymbol{1}_S,5}$. Now, we move to the next sequence of hybrids for $t = T+1$ as depicted in Table 5.

Hybrid $\widetilde{\mathsf{H}}_3$. It is identical to $\mathsf{H}_{3,T,N,S,\boldsymbol{1}_S,5}$ except the position sim of $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$ is zeroed out and $\widetilde{\boldsymbol{v}}_{k,q}[\mathsf{sim}]$ is set to $\boldsymbol{s}_{k,f}[q]$ for all $k \in \mathcal{I}_{\boldsymbol{M}}$. The inner products between the vectors are

Table 6 The outer loop hybrids running from $t = 1$ to T in the security proof of 1-UAWS for the case where the ciphertext challenge comes after the secret key query

hybrid	vector	$\mathsf{rand}, \mathsf{tb}_{\mathcal{T}}$	$\mathsf{rand}^{\mathsf{temp}}, \mathsf{tb}_{\mathcal{T}}^{\mathsf{temp}}$	sim	$\mathsf{sim}^{\mathsf{temp}}$
$\mathsf{H}_{3,t,1}$	$\boldsymbol{v}_{k,q}$	normal	$0, 0$	$\boldsymbol{s}_{k,f}[q]$	0
	$\boldsymbol{u}_{k,t'<t,i,j,\boldsymbol{W}}$	$0, 0$	$0, 0$	$\boldsymbol{s}_{\boldsymbol{x}}[(t', i, j, \boldsymbol{W})]$	0
	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$	normal	$0, 0$	0	0
	$\boldsymbol{u}_{k,t'>t,i,j,\boldsymbol{W}}$	normal	$0, 0$	0	0
$\mathsf{H}_{3,t,2} \equiv \mathsf{H}_{3,t,3,1,,1}$	$\boldsymbol{v}_{k,q}$	normal	$\boxed{\text{normal}}$	$\boldsymbol{s}_{k,f}[q]$	0
	$\boldsymbol{u}_{k,t'<t,i,j,\boldsymbol{W}}$	$0, 0$	$0, 0$	$\boldsymbol{s}_{\boldsymbol{x}}[(t', i, j, \boldsymbol{W})]$	0
	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$	$\boxed{0, 0}$	$\boxed{\text{normal}}$	0	$\boxed{\boldsymbol{s}_{\boldsymbol{x}}[(t, i, j, \boldsymbol{W})]}$
	$\boldsymbol{u}_{k,t'>t,i,j,\boldsymbol{W}}$	normal	$0, 0$	0	0
$\mathsf{H}_{3,t,3,1\sim Q,1\sim 5}$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\mathsf{H}_{3,t,4} \equiv \mathsf{H}_{3,t,3,Q,5}$	$\boldsymbol{v}_{k,q}$	normal	$\boxed{0, 0}$	$\boldsymbol{s}_{k,f}[q]$	$\boxed{\boldsymbol{s}_{k,f}[q]}$
	$\boldsymbol{u}_{k,t'<t,i,j,\boldsymbol{W}}$	$0, 0$	$0, 0$	$\boldsymbol{s}_{\boldsymbol{x}}[(t', i, j, \boldsymbol{W})]$	0
	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$	$0, 0$	normal	0	$\boldsymbol{s}_{\boldsymbol{x}}[(t, i, j, \boldsymbol{W})]$
	$\boldsymbol{u}_{k,t'>t,i,j,\boldsymbol{W}}$	normal	$0, 0$	0	0
$\mathsf{H}_{3,t,5} \equiv \mathsf{H}_{3,t+1,1}$	$\boldsymbol{v}_{k,q}$	normal	$0, 0$	$\boldsymbol{s}_{k,f}[q]$	$\boxed{0}$
	$\boldsymbol{u}_{k,t'<t,i,j,\boldsymbol{W}}$	$0, 0$	$0, 0$	$\boldsymbol{s}_{\boldsymbol{x}}[(t', i, j, \boldsymbol{W})]$	0
	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$	$0, 0$	$\boxed{0, 0}$	$\boxed{\boldsymbol{s}_{\boldsymbol{x}}[(t, i, j, \boldsymbol{W})]}$	$\boxed{0}$
	$\boldsymbol{u}_{k,t'>t,i,j,\boldsymbol{W}}$	normal	$0, 0$	0	0

For brevity, $\boldsymbol{u}_{\mathsf{init}}, \boldsymbol{v}_{\mathsf{init}}, \widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}, \widetilde{\boldsymbol{v}}_{k,q}, \boldsymbol{v}_{k,q}[\mathsf{acc}] = 0, \boldsymbol{u}_{k,t\leq T,i,j,\boldsymbol{W}}[\mathsf{acc}] = 0$ are suppressed.

The reversely sampled $\ell_{k,\mathsf{init}}$ is hardwired in $\boldsymbol{u}_{k,\mathsf{init}}$:

$$\ell_{k,\mathsf{init}} \leftarrow \mathsf{RevSamp}((M_k, 1^N, 1^T, 1^{2^S}), \boldsymbol{x}, \boldsymbol{z}[k]M_k(\boldsymbol{x}) + \beta_k, (\ell_{k,t,\theta_k})_{t\in[T+1],\theta_k\in\mathcal{C}_{M_k,N,S}})$$

The "normal" in	$\mathsf{rand}, \mathsf{rand}^{\mathsf{temp}}$	$\mathsf{tb}_{\mathcal{T}}, \mathsf{tb}_{\mathcal{T}}^{\mathsf{temp}}$
$\boldsymbol{v}_{k,q}$:	$-\boldsymbol{r}_{k,f}[q]$	$(\mathbf{M}_{k,\mathcal{T}}\boldsymbol{r}_{k,f})[q]$
$t \leq T$, $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$:	$\boldsymbol{r}_{\boldsymbol{x}}[(t-1, i, j, \boldsymbol{W})]$	$c_{\mathcal{T}}(\boldsymbol{x}, t, i, j, \boldsymbol{W}; \boldsymbol{r}_{\boldsymbol{x}})$

In this iteration, the labels $\ell_{k,t,i,j,\boldsymbol{W},q}$ with t are computed as:

$t < t'$:	simulated as $\boldsymbol{s}_{\boldsymbol{x}}[(t', i, j, \boldsymbol{W})]\boldsymbol{s}_{k,f}[q]$ and computed using the slot sim
$t = t'$:	computed honestly using IPFE in $\widetilde{\mathsf{H}}_{3,t,1}$
	computed honestly via temporary slots $\mathsf{rand}^{\mathsf{temp}}, \mathsf{tb}_{\mathcal{T}}^{\mathsf{temp}}$ in $\widetilde{\mathsf{H}}_{3,t,2}$
	simulated and computed using the slot $\mathsf{sim}^{\mathsf{temp}}$ in $\mathsf{H}_{3,t,4}$
	simulated and computed using the slot sim in $\mathsf{H}_{3,t,5}$
$t' > t$:	computed honestly using IPFE

unchanged in $\mathsf{H}_{3,T,N,S,\mathbf{1}_S,5}$ and $\widetilde{\mathsf{H}}_3$. Thus, the indistinguishability between these two hybrids is ensured by the function security of IPFE.

Hybrid $\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},1}$. It proceeds identically to $\widetilde{\mathsf{H}}_3$ except that for all $(i', j', \boldsymbol{W}') < (i, j, \boldsymbol{W})$, $\widetilde{\boldsymbol{u}}_{k,T+1,i',j',\boldsymbol{W}'}$ has its values in rand and acc's cleared, and that a random value $\boldsymbol{s}_{\boldsymbol{x}}[(T + 1, i', j', \boldsymbol{W}')]$ is embedded in $\widetilde{\boldsymbol{u}}_{k,T+1,i',j',\boldsymbol{W}'}[\mathsf{sim}]$.

Hybrid $\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},2}$. It proceeds exactly the same way as $\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},1}$ except that the values in $\widetilde{\boldsymbol{u}}_{k,t,i,j,\boldsymbol{W}}$ are set to zero and its inner product with $\widetilde{\boldsymbol{v}}_{k,q}$'s, i.e. the labels $\ell_{k,T+1,i,j,\boldsymbol{W},q}$ for all k, q, are hardcoded into $\widetilde{\boldsymbol{v}}_{k,q}$'s as follows:

- The positions rand and acc of $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$ are set to 0.
- The value at $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}[\mathsf{sim}^{\mathsf{temp}}]$ is set to 1.
- The honest labels $\ell_{k,T+1,i,j,\boldsymbol{W},q} = -\boldsymbol{r}_{\boldsymbol{x}}[(T,i,j,\boldsymbol{W})]\boldsymbol{r}_{k,f}[q] + \cdots$ are embedded in $\widetilde{\boldsymbol{v}}_{k,q}[\mathsf{sim}^{\mathsf{temp}}]$ for each $q \in [Q]$ and $k \in \mathcal{I}_{\boldsymbol{M}}$ where "$\cdots$" represents the term $\boldsymbol{y}_k[q]\boldsymbol{z}[k]$.

The inner products between the vectors are unchanged, and hence the indistinguishability between the hybrids $\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},1}$ and $\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},2}$ is guaranteed by the function hiding security of IPFE.

Hybrid $\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},3}$. It proceeds similar to $\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},2}$ except that the labels $\ell_{k,T+1,i,j,\boldsymbol{W},q}$ are changed to truly randomized values. We can invoke DDH assumption in $\mathbb{G}_2$ as before to show the indistinguishability between the hybrids $\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},2}$ and $\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},3}$ since the random values $\boldsymbol{r}_{\boldsymbol{x}}[(T,i,j,\boldsymbol{W})]$ and $\boldsymbol{r}_{k,f}[q]$'s only appear in the exponent of $\mathbb{G}_2$ and hence the label functions can be truly randomized due to the special piecewise security of AKGS. Note that, the values $[\![\ell_{k,\mathsf{init}}]\!]_2 \leftarrow \mathsf{RevSamp}(\cdots)$ can be efficiently computed in the exponent of $\mathbb{G}_2$.

Hybrid $\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},4}$. It proceeds identical to $\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},3}$ except the truly random labels $\ell_{k,T+1,i,j,\boldsymbol{W},q}$ for all $q \in [Q], k \in \mathcal{I}_{\boldsymbol{M}}$ are replaced by pseudorandom values $\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]\boldsymbol{s}_{k,f}[q]$. The hybrids $\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},3}$ and $\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},4}$ are indistinguishable due to the DDH assumption in $\mathbb{G}_2$.

Hybrid $\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},5}$. It proceeds exactly the same way as $\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},4}$ except the pseudorandom labels $\ell_{k,T+1,i,j,\boldsymbol{W},q} = \boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]\boldsymbol{s}_{k,f}[q]$ hardwired in $\widetilde{\boldsymbol{v}}_{k,q}[\mathsf{sim}^{\mathsf{temp}}]$'s are *split* into $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}[\mathsf{sim}]$ (embedding the factor $\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$) and $\widetilde{\boldsymbol{v}}_{k,q}[\mathsf{sim}]$'s (embedding the factor $\boldsymbol{s}_{k,f}[q]$). The inner products in the hybrids $\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},4}$ and $\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},5}$ are unchanged and hence the these two hybrids are indistinguishable due to the function hiding security of IPFE. Moreover, it can be observed that $\widetilde{\mathsf{H}}_{3,i,j,\boldsymbol{W},5} \equiv \widetilde{\mathsf{H}}_{3,i',j',\boldsymbol{W}',3}$ for $(i',j',\boldsymbol{W}') = (i,j,\boldsymbol{W}) + 1$.

Therefore, in this sequence of hybrids for $t = T+1$, we have $\widetilde{\mathsf{H}}_{3,1,1,\boldsymbol{0}_S,1} \approx \widetilde{\mathsf{H}}_{3,N,S,\boldsymbol{1}_S,5}$. Lastly, we observe that $\widetilde{\mathsf{H}}_{3,N,S,\boldsymbol{1}_S,5} \equiv \mathsf{H}_4$ and hence $\mathsf{H}_2 \approx \mathsf{H}_4$ for the case of CT before SK. □

Proof of Claim 2 The case of SK before CT for showing $\mathsf{H}_2 \approx \mathsf{H}_4$ is more involved and further difficulties arises since we have *two independent* IPFEs for each Turing machine in contrast to the security analysis of [62] where only a single IPFE was sufficient.

The overall goal of the claim is to make all the label values $\ell_{k,t,i,j,\boldsymbol{W},q}$ simulated by invoking DDH similar to the case of CT before SK. However, since the secret key comes before the challenge ciphertext and $\ell_{k,\mathsf{init}} \leftarrow \mathsf{RevSamp}(\cdots)$ is computed while encryption, we can only apply DDH into the ciphertext vectors which are computed in the exponent of $\mathbb{G}_1$. Thus, we have to move $\boldsymbol{r}_{k,f}[q]$ into the ciphertext vectors (Table 7). But, in this case, $\boldsymbol{r}_{k,f}[q]$ of $\boldsymbol{v}_{k,q}$ may appear in $(\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q']$ of any $\boldsymbol{v}_{k,q'}$ depending on the transition block. Moreover, $\boldsymbol{r}_{k,f}[q]$ also presents in $\widetilde{\boldsymbol{v}}_{k,q}$ which are associated to *second* IPFE. Hence, in the security analysis, we must take care of the following facts:

- The special piecewise security can only be applied in the increasing order of t for changing $\ell_{k,t,i,j,\boldsymbol{W},q}$'s to their simulated form.
- More importantly, to simulate $\ell_{k,t,i,j,\boldsymbol{W},q}$ for $t \le T$, all occurrence of $\boldsymbol{r}_{k,f}[q]$ must be in the ciphertext of *both* the IPFE. Also, we can not simulate $\ell_{k,T+1,i,j,\boldsymbol{W},q}$ (in the *second* IPFE) while simulating $\ell_{k,t',i,j,\boldsymbol{W},q}$ (in the *first* IPFE).
- There is not enough space in the ciphertext to embed all the $\boldsymbol{r}_{k,f}[q]$'s at the same time for each $k \in \mathcal{I}_{\boldsymbol{M}}$.
- The values $\boldsymbol{r}_{k,f}[q]$ must *not* go away until *all* $\ell_{k,t,i,j,\boldsymbol{W},q}$'s are simulated. Indeed, $\boldsymbol{r}_{k,f}[q]$ still resides in $\boldsymbol{v}_{k,q'}$'s in H_4, the end hybrid of the claim.

Table 7 The inner loop hybrids in the security proof of 1-UAWS for the case where the ciphertext challenge comes after the secret key query

hybrid	vector	rand, $\text{rand}^{\text{comp}}$ tb_τ, $\text{tb}_\tau^{\text{comp}}$	$\text{rand}^{\text{temp}}$, $\text{rand}^{\text{temp,comp}}$, $\text{tb}_\tau^{\text{temp}}$, $\text{tb}_\tau^{\text{temp,comp}}$	sim	sim^{temp}	sim^{comp}
$\mathsf{H}_{3,t,3,q,1}$	$\boldsymbol{v}_{k,q'<q}$	normal	$0,0,0,0$	$\boldsymbol{s}_{k,f}[q']$	$\boldsymbol{s}_{k,f}[q']$	0
	$\boldsymbol{v}_{k,q}$	normal	normal	$\boldsymbol{s}_{k,f}[q]$	0	0
	$\boldsymbol{v}_{k,q'>q}$	normal	normal	$\boldsymbol{s}_{k,f}[q']$	0	0
	$\boldsymbol{u}_{k,t'<t,i,j,\boldsymbol{W}}$	$0,0,0,0$	$0,0,0,0$	$\boldsymbol{s}_{\boldsymbol{x}}[(t',i,j,\boldsymbol{W})]$	0	0
	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$	$0,0,0,0$	normal	0	$\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]$	0
	$\boldsymbol{u}_{k,t'>t,i,j,\boldsymbol{W}}$	normal	$0,0,0,0$	0	0	0
		rand, acc	sim	sim^{temp}		
	$\widetilde{\boldsymbol{v}}_{k,q'<q}$	normal	0	0		
	$\widetilde{\boldsymbol{v}}_{k,q}$	normal	0	0		
	$\widetilde{\boldsymbol{v}}_{k,q'>q}$	normal	0	0		
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	normal	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$	0		
$\mathsf{H}_{3,t,3,q,2}$	$\boldsymbol{v}_{k,q'<q}$	$\boxed{\text{✗}\boldsymbol{r}_{k,f}[q]}$	$0,0,0,0$	$\boldsymbol{s}_{k,f}[q']$	$\boldsymbol{s}_{k,f}[q']$	0
	$\boldsymbol{v}_{k,q}$	$\boxed{\text{✗}\boldsymbol{r}_{k,f}[q]}$	$\boxed{0,0,0,0}$	$\boxed{0}$	0	$\boxed{1}$
	$\boldsymbol{v}_{k,q'>q}$	$\boxed{\text{✗}\boldsymbol{r}_{k,f}[q]}$	$\boxed{\text{✗}\boldsymbol{r}_{k,f}[q]}$	$\boldsymbol{s}_{k,f}[q']$	0	0
	$\boldsymbol{u}_{k,t'<t,i,j,\boldsymbol{W}}$	$0,0,0,0$	$0,0,0,0$	$\boldsymbol{s}_{\boldsymbol{x}}[(t',i,j,\boldsymbol{W})]$	0	$\boxed{\boldsymbol{s}_{\boldsymbol{x}}[(t',i,j,\boldsymbol{W})]\boldsymbol{s}_{k,f}[q]}$
	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$	$0,0,0,0$	$\boxed{\text{✓}\boldsymbol{r}_{k,f}[q]}$	0	$\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]$	$\boxed{\begin{array}{l}\text{honest } \ell_{k,t,i,j,\boldsymbol{W},q}\\ = -\boldsymbol{r}_{\boldsymbol{x}}[(t-1,i,j,\boldsymbol{W})]\boldsymbol{r}_{k,f}[q] + \cdots\end{array}}$
	$\boldsymbol{u}_{k,t'>t,i,j,\boldsymbol{W}}$	$\boxed{\text{✓}\boldsymbol{r}_{k,f}[q]}$	$0,0,0,0$	0	0	0
		rand, acc	sim	sim^{temp}		
	$\widetilde{\boldsymbol{v}}_{k,q'<q}$	normal	0	0		
	$\widetilde{\boldsymbol{v}}_{k,q}$	$\boxed{0,0}$	0	$\boxed{1}$		
	$\widetilde{\boldsymbol{v}}_{k,q'>q}$	normal	0	0		
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	normal	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$	$\boxed{\begin{array}{l}\text{honest } \ell_{k,T+1,i,j,\boldsymbol{W},q}\\ = -\boldsymbol{r}_{\boldsymbol{x}}[(T,i,j,\boldsymbol{W})]\boldsymbol{r}_{k,f}[q] + \cdots\end{array}}$		
$\mathsf{H}_{3,t,3,q,3}$	$\boldsymbol{v}_{k,q'<q}$	$\text{✗}\boldsymbol{r}_{k,f}[q]$	$0,0,0,0$	$\boldsymbol{s}_{k,f}[q']$	$\boldsymbol{s}_{k,f}[q']$	0
	$\boldsymbol{v}_{k,q}$	$\text{✗}\boldsymbol{r}_{k,f}[q]$	$0,0,0,0$	0	0	1
	$\boldsymbol{v}_{k,q'>q}$	$\text{✗}\boldsymbol{r}_{k,f}[q]$	$\text{✗}\boldsymbol{r}_{k,f}[q]$	$\boldsymbol{s}_{k,f}[q']$	0	0
	$\boldsymbol{u}_{k,t'<t,i,j,\boldsymbol{W}}$	$0,0,0,0$	$0,0,0,0$	$\boldsymbol{s}_{\boldsymbol{x}}[(t',i,j,\boldsymbol{W})]$	0	$\boldsymbol{s}_{\boldsymbol{x}}[(t',i,j,\boldsymbol{W})]\boldsymbol{s}_{k,f}[q]$
	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$	$0,0,0,0$	$\text{✓}\boldsymbol{r}_{k,f}[q]$	0	$\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]$	$\boxed{\ell_{k,t,i,j,\boldsymbol{W},q} \xleftarrow{\$} \mathbb{Z}_p}$
	$\boldsymbol{u}_{k,t'>t,i,j,\boldsymbol{W}}$	$\text{✓}\boldsymbol{r}_{k,f}[q]$	$0,0,0,0$	0	0	0
		rand, acc	sim	sim^{temp}		
	$\widetilde{\boldsymbol{v}}_{k,q'<q}$	normal	0	0		
	$\widetilde{\boldsymbol{v}}_{k,q}$	$0,0$	0	1		
	$\widetilde{\boldsymbol{v}}_{k,q'>q}$	normal	0	0		
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	normal	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$	honest $\ell_{k,T+1,i,j,\boldsymbol{W},q} = -\boldsymbol{r}_{\boldsymbol{x}}[(T,i,j,\boldsymbol{W})]\boldsymbol{r}_{k,f}[q] + \cdots$		

To deal with all these facts, we employ a strategy inspired from the proof technique of [62] where they use a two-level loop over t, q *with* $t \le T$ and switch, in the increasing order of t, q, batches of $NS2^S$ label functions. That is, for fixed t, q and all $i, j, \boldsymbol{W}$ and for all $k \in \mathcal{I}_{\boldsymbol{M}}$, the batches of label values $\ell_{k,t,i,j,\boldsymbol{W},q}$ are simulated by moving $\boldsymbol{r}_{k,f}[q]$'s back and forth in each iteration. More precisely, in each iteration of t, q, when moving $\boldsymbol{r}_{k,f}[q]$ into the ciphertext vectors, we erase all its occurrence in the secret key vectors of *both* the IPFE and must *compensate* some $\ell_{t',i,j,\boldsymbol{W},q'}$'s for their loss of $\boldsymbol{r}_{k,f}[q]$ using the indices with superscript comp in the case of $t' \le T$. Observe that, $\boldsymbol{r}_{k,f}[q]$ only appears in the position rand of $\widetilde{\boldsymbol{v}}_{k,q}$ of the *second* IPFE. Thus, it is *not* required to *compensate* the loss of $\boldsymbol{r}_{k,f}[q]$ in any other $\ell_{T+1,i,j,\boldsymbol{W},q'}$'s. However, $\boldsymbol{r}_{k,f}[q]$ is still required to shift into the ciphertext vectors of the second IPFE. We use the indices with superscript temp to hardcode the honest label values of $\ell_{T+1,i,j,\boldsymbol{W},q}$ while running the loop over t, q with $t \le T$. Finally, after the two-level loop running over t, q with $t \le T$ ends, we *erase* $\boldsymbol{r}_{k,f}[q]$ from $\boldsymbol{v}_{k,q}$ and run a separate loop over q

Table 8 The inner loop hybrids in the security proof of 1-UAWS for the case where the ciphertext challenge comes after the secret key query

hybrid	vector	$\mathsf{rand}, \mathsf{rand}^{\mathsf{comp}}$, $\mathsf{tb}_\tau, \mathsf{tb}_\tau^{\mathsf{comp}}$	$\mathsf{rand}^{\mathsf{temp}}, \mathsf{rand}^{\mathsf{temp,comp}}$, $\mathsf{tb}_\tau^{\mathsf{temp}}, \mathsf{tb}_\tau^{\mathsf{temp,comp}}$	sim	$\mathsf{sim}^{\mathsf{temp}}$	$\mathsf{sim}^{\mathsf{comp}}$
$\mathsf{H}_{3,t,3,q,4}$	$\boldsymbol{v}_{k,q'<q}$	$✗\boldsymbol{r}_{k,f}[q]$	$0,0,0,0$	$\boldsymbol{s}_{k,f}[q']$	$\boldsymbol{s}_{k,f}[q']$	0
	$\boldsymbol{v}_{k,q}$	$✗\boldsymbol{r}_{k,f}[q]$	$0,0,0,0$	0	0	1
	$\boldsymbol{v}_{k,q'>q}$	$✗\boldsymbol{r}_{k,f}[q]$	$✗\boldsymbol{r}_{k,f}[q]$	$\boldsymbol{s}_{k,f}[q']$	0	0
	$\boldsymbol{u}_{k,t'<t,i,j,\boldsymbol{W}}$	$0,0,0,0$	$0,0,0,0$	$\boldsymbol{s}_{\boldsymbol{x}}[(t',i,j,\boldsymbol{W})]$	0	$\boldsymbol{s}_{\boldsymbol{x}}[(t',i,j,\boldsymbol{W})]\boldsymbol{s}_{k,f}[q]$
	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$	$0,0,0,0$	$✓\boldsymbol{r}_{k,f}[q]$	0	$\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]$	$\boxed{\text{simulated } \ell_{k,t,i,j,\boldsymbol{W},q} = \boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]\boldsymbol{s}_{k,f}[q]}$
	$\boldsymbol{u}_{k,t'>t,i,j,\boldsymbol{W}}$	$✓\boldsymbol{r}_{k,f}[q]$	$0,0,0,0$	0	0	0
		$\mathsf{rand}, \mathsf{acc}$	sim	$\mathsf{sim}^{\mathsf{temp}}$		
	$\widetilde{\boldsymbol{v}}_{k,q'<q}$	normal	0	0		
	$\widetilde{\boldsymbol{v}}_{k,q}$	$0,0$	0	1		
	$\widetilde{\boldsymbol{v}}_{k,q'>q}$	normal	0	0		
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	normal	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$	honest $\ell_{k,T+1,i,j,\boldsymbol{W},q} = -\boldsymbol{r}_{\boldsymbol{x}}[(T,i,j,\boldsymbol{W})]\boldsymbol{r}_{k,f}[q] + \cdots$		
$\mathsf{H}_{3,t,3,q,5} \equiv \mathsf{H}_{3,t,3,q+1,1}$	$\boldsymbol{v}_{k,q'<q}$	$\boxed{\text{normal}}$	$0,0,0,0$	$\boldsymbol{s}_{k,f}[q']$	$\boldsymbol{s}_{k,f}[q']$	0
	$\boldsymbol{v}_{k,q}$	$\boxed{\text{normal}}$	$0,0,0,0$	$\boxed{\boldsymbol{s}_{k,f}[q]}$	$\boxed{\boldsymbol{s}_{k,f}[q]}$	$\boxed{0}$
	$\boldsymbol{v}_{k,q'>q}$	$\boxed{\text{normal}}$	$\boxed{\text{normal}}$	$\boldsymbol{s}_{k,f}[q']$	0	0
	$\boldsymbol{u}_{k,t'<t,i,j,\boldsymbol{W}}$	$0,0,0,0$	$0,0,0,0$	$\boldsymbol{s}_{\boldsymbol{x}}[(t',i,j,\boldsymbol{W})]$	0	$\boxed{0}$
	$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$	$0,0,0,0$	$\boxed{\text{normal}}$	0	$\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]$	$\boxed{0}$
	$\boldsymbol{u}_{k,t'>t,i,j,\boldsymbol{W}}$	$\boxed{\text{normal}}$	$0,0,0,0$	0	0	0
		$\mathsf{rand}, \mathsf{acc}$	sim	$\mathsf{sim}^{\mathsf{temp}}$		
	$\widetilde{\boldsymbol{v}}_{k,q'<q}$	normal	0	0		
	$\widetilde{\boldsymbol{v}}_{k,q}$	$\boxed{\text{normal}}$	0	$\boxed{0}$		
	$\widetilde{\boldsymbol{v}}_{k,q'>q}$	normal	0	0		
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	normal	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$	$\boxed{0}$		

For brevity, $\boldsymbol{u}_{\mathsf{init}}, \boldsymbol{v}_{\mathsf{init}}, \boldsymbol{v}_{k,q}[\mathsf{acc}] = 0, \boldsymbol{u}_{k,t\le T,i,j,\boldsymbol{W}}[\mathsf{acc}] = 0$ are suppressed.

For brevity, $\boldsymbol{u}_{\mathsf{init}}, \boldsymbol{v}_{\mathsf{init}}, \boldsymbol{v}_{k,q}[\mathsf{acc}] = 0, \boldsymbol{u}_{k,t\le T,i,j,\boldsymbol{W}}[\mathsf{acc}] = 0$ are suppressed. The reversely sampled $\ell_{k,\mathsf{init}}$ is hardwired in $\boldsymbol{u}_{k,\mathsf{init}}$, and is only needed (and can only be computed so by the reduction) in the exponent of G_1:

$$[\![\ell_{k,\mathsf{init}}]\!]_1 \leftarrow \mathsf{RevSamp}((M_k, 1^N, 1^T, 1^{2^S}), \boldsymbol{x}, [\![\boldsymbol{z}[k]M_k(\boldsymbol{x}) + \beta_k]\!]_1, ([\![\ell_{k,t,\theta_k}]\!]_1)_{t\in[T+1],\theta_k\in\mathcal{C}_{M_k,N,S}}).$$

The omitted term "$\cdots$": $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}[\mathsf{sim}^{\mathsf{comp}}] = (\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q]$ and $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}[\mathsf{sim}^{\mathsf{temp}}] = \boldsymbol{y}_k[q]\boldsymbol{z}[k]$.

The "normal" in	$\mathsf{rand}, \mathsf{rand}^{\mathsf{temp}}$	$\mathsf{tb}_\tau, \mathsf{tb}_\tau^{\mathsf{temp}}$	$\mathsf{rand}^{\mathsf{comp}}, \mathsf{rand}^{\mathsf{temp,comp}}$, $\mathsf{tb}_\tau^{\mathsf{comp}}, \mathsf{tb}_\tau^{\mathsf{temp,comp}}$
$\boldsymbol{v}_{k,q}$:	$-\boldsymbol{r}_{k,f}[q]$	$(\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q]$	0
$t' \le T,\ \boldsymbol{u}_{k,t',i,j,\boldsymbol{W}}$:	$\boldsymbol{r}_{\boldsymbol{x}}[(t'-1,i,j,\boldsymbol{W})]$	$c_\tau(\boldsymbol{x}, t', i, j, \boldsymbol{W}; \boldsymbol{r}_{\boldsymbol{x}})$	0

The "normal" in $\widetilde{\boldsymbol{v}}_{k,q}[\mathsf{rand}] = -\boldsymbol{r}_{k,f}[q]$, $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}[\mathsf{rand}] = \boldsymbol{r}_{\boldsymbol{x}}[(T,i,j,\boldsymbol{W})]$
$\widetilde{\boldsymbol{v}}_{k,q}[\mathsf{acc}] = \boldsymbol{y}_k[q]$, $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}[\mathsf{acc}] = \boldsymbol{z}[k]$

in the increasing order to simulate the labels $\ell_{T+1,i,j,\boldsymbol{W},q}$'s using the indices with superscript temp in the second IPFE.

We define *modes* of a label $\ell_{t',i,j,\boldsymbol{W},q'}$ for ease of understanding the loops used in this claim (Table 9). The definitions of *modes* are similar to what used by [62]. There are three orthogonal group of nodes:

- The first group is about the value of the label. A label is *honest* if its value $L_{t',i,j,\boldsymbol{W},q'}(\boldsymbol{x})$ is computed using the garbling randomness $\boldsymbol{r} = \boldsymbol{r}_{\boldsymbol{x}} \otimes \boldsymbol{r}_f$. It is *random* if its value is sampled uniformly at random. It is *simulated* if its value is $\boldsymbol{s}_{\boldsymbol{x}}[(t', i, j, \boldsymbol{W})]\boldsymbol{s}_{k,f}[q']$.
- The second group is about where the terms $\boldsymbol{r}_f$ and $\boldsymbol{s}_f$ are placed while computing the labels using the IPFEs. A label is normal (this is the default) if $\boldsymbol{r}_f, \boldsymbol{s}_f$ are placed in the secret key. It is *compensated* if $\boldsymbol{r}_f[q], \boldsymbol{s}_f[q]$ are placed in the ciphertext while the other components of $\boldsymbol{r}_f, \boldsymbol{s}_f$ are still in the secret key (for simplicity, we note that this mode only appears in the *first* IPFE). It is *hardwired* if the value (in its entirety) is hardwired

Table 9 Table 5: the remaining notes

The compensation ($\text{✗}\boldsymbol{r}_{k,f}[q], \text{✓}\boldsymbol{r}_{k,f}[q]$) components in ...

	$\mathsf{rand}^{?}$	$\mathsf{rand}^{?,\mathsf{comp}}$	$\mathsf{tb}_{\tau}^{?}$	$\mathsf{tb}_{\tau}^{?,\mathsf{comp}}$
$q' \neq q,\ \boldsymbol{v}_{k,q'}$:	$-\boldsymbol{r}_{k,f}[q']$	0	$(\mathbf{M}_{k,\tau}(\boldsymbol{r}_{k,f} - \boldsymbol{r}_{k,f}[q]\boldsymbol{e}_q))[q']$	$(\mathbf{M}_{k,\tau}\boldsymbol{e}_q)[q']$
$\boldsymbol{v}_{k,q}$:	0	-1	$(\mathbf{M}_{k,\tau}(\boldsymbol{r}_{k,f} - \boldsymbol{r}_{k,f}[q]\boldsymbol{e}_q))[q]$	$(\mathbf{M}_{k,\tau}\boldsymbol{e}_q)[q]$
$t' < t,\ \boldsymbol{u}_{k,t',i,j,\boldsymbol{W}}$:	$\boldsymbol{r}_{\boldsymbol{x}}[(t'-1,i,j,\boldsymbol{W})]$	$\boldsymbol{r}_{\boldsymbol{x}}[(t'-1,i,j,\boldsymbol{W})]\boldsymbol{r}_{k,f}[q]$	$c_{\mathcal{T}}(\boldsymbol{x},t',i,j,\boldsymbol{W};\boldsymbol{r}_{\boldsymbol{x}})$	$c_{\mathcal{T}}(\boldsymbol{x},t',i,j,\boldsymbol{W};\boldsymbol{r}_{\boldsymbol{x}})\boldsymbol{r}_{k,f}[q]$
$t \leq T,\ \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$:	$\boldsymbol{r}_{\boldsymbol{x}}[(t'-1,i,j,\boldsymbol{W})]$	0	$c_{\mathcal{T}}(\boldsymbol{x},t,i,j,\boldsymbol{W};\boldsymbol{r}_{\boldsymbol{x}})$	$c_{\mathcal{T}}(\boldsymbol{x},t,i,j,\boldsymbol{W};\boldsymbol{r}_{\boldsymbol{x}})\boldsymbol{r}_{k,f}[q]$

In the above table, "?" is either nothing or "temp", i.e., if the values are set in both non-temporary and temporary slots, they are the same. Note that, $\boldsymbol{r}_{k,f} - \boldsymbol{r}_{k,f}[q]\boldsymbol{e}_q$ is simply $\boldsymbol{r}_{k,f}$ with its q^{th} entry changed to 0, whence $\boldsymbol{r}_{k,f}[q]$ does not appear. The compensation is governed by the following identity for $t' \leq T$:

$$\begin{aligned}\ell_{t',i,j,\boldsymbol{W},q'} &= \boldsymbol{r}_{\boldsymbol{x}}[(t'-1,i,j,\boldsymbol{W})]\boldsymbol{r}_{k,f}[q'] \quad + \textstyle\sum_{\tau\in\mathcal{T}} c_{\mathcal{T}}(\boldsymbol{x},t',i,j,\boldsymbol{W};\boldsymbol{r}_{\boldsymbol{x}})(\mathbf{M}_{k,\tau}(\boldsymbol{r}_{k,f} - \boldsymbol{r}_{k,f}[q]\boldsymbol{e}_q + \boldsymbol{r}_{k,f}[q]\boldsymbol{e}_q))[q] \\ &= \boldsymbol{r}_{\boldsymbol{x}}[(t'-1,i,j,\boldsymbol{W})]\boldsymbol{r}_{k,f}[q'] \quad + \textstyle\sum_{\tau\in\mathcal{T}} c_{\mathcal{T}}(\boldsymbol{x},t',i,j,\boldsymbol{W};\boldsymbol{r}_{\boldsymbol{x}})(\mathbf{M}_{k,\tau}(\boldsymbol{r}_{k,f} - \boldsymbol{r}_{k,f}[q]\boldsymbol{e}_q)[q] \\ &\qquad + \sum_{\tau\in\mathcal{T}} c_{\mathcal{T}}(\boldsymbol{x},t',i,j,\boldsymbol{W};\boldsymbol{r}_{\boldsymbol{x}})\boldsymbol{r}_{k,f}[q]\cdot(\mathbf{M}_{k,\tau}\boldsymbol{e}_q)[q'] \qquad (4)\end{aligned}$$

In this iteration, the labels $\ell_{k,t',i,j,\boldsymbol{W},q'}$ with (t',q') are computed as:

	$q' < q$	$q' = q$	$q' > q$
$t' < t \leq T$:	S	S $\to$ SC $\to$ S	S
$t' = t \leq T$:	ST	$\boxed{\text{HT} \to \text{HW} \to \text{RW} \to \text{SW} \to \text{ST}}$	HT $\to$ HCT $\to$ HT
$T \geq t' > t$:	H $\to$ HC $\to$ H	H $\to$ HC $\to$ H	H $\to$ HC $\to$ H
$t' = T+1$:	H	$\boxed{\text{H} \to \text{HW} \to \text{HW} \to \text{HW} \to \text{H}}$	H

The shorthands are **H**onest, **R**andom, **S**imulated, **C**ompensated, hard**W**ired, **T**emporary.

The net effect is that $\ell_{k,t\leq T,i,j,\boldsymbol{W},q}$'s change from *honest and temporary* to *simulated and temporary.*

Note that, in this iteration, $\ell_{k,T+1,i,j,\boldsymbol{W},q'}$'s are unchanged for all q'.

The value $\ell_{k,T+1,i,j,\boldsymbol{W},q'=q}$ is *honest and hard wired* in the intermediate hybrids $\mathsf{H}_{3,t,3,q,2\sim 4}$.

in the ciphertext (for simplicity, we note that this mode only appears to the labels with $t' = t, q' = q$).

– In the last group, a label is normal (default) if it is computed without indices with superscript temp. It is *temporary* if it is computed with indices having superscript temp.

As discussed above, the first loop of this claim is a *two-level* loop with outer loop running over $t = 1, \ldots, T$ (provided in Table 6) and the inner loop running over $q = 1, \ldots, Q$ (given in Table 8). We call this *part 1* of the proof. The second loop runs over $q = 1, \ldots, Q$ (described in Table 10) and it is dedicated for simulating the label values $\ell_{k,T+1,i,j,\boldsymbol{W},q}$ for all $k \in \mathcal{I}_{\boldsymbol{M}}$.We call this *part 2* of the proof. In this hybrids, the secret key vectors $\boldsymbol{v}$'s appear before the ciphertext vectors $\boldsymbol{u}$'s.

□

Part 1 The sequence of hybrids in the two-level loop (with $t \leq T, q \leq Q$) and their indistinguishability arguments (Table 11).

Hybrid $\mathsf{H}_{3,t,1}$. It proceeds identically to H_2 except that for all $t' < t \leq T$ and all $i, j, \boldsymbol{W}$, the vectors $\boldsymbol{u}_{k,t',i,j,\boldsymbol{W}}$ have their values at rand and tb_{τ}'s cleared, and that a random value $\boldsymbol{s}_{\boldsymbol{x}}[(t', i, j, \boldsymbol{W})]$ is embedded in $\boldsymbol{u}_{k,t',i,j,\boldsymbol{W}}[\mathsf{sim}]$. This means that all the labels for $(t < t' \leq T, i, j, \boldsymbol{W})$ are simulated, the first label $\ell_{k,\mathsf{init}}$ is reversely sampled and the rest are honestly computed.

Hybrid $\mathsf{H}_{3,t,2}$. It proceeds exactly the same way as $\mathsf{H}_{3,t,1}$ except that the modes of $\ell_{k,t,i,j,\boldsymbol{W},q}$'s (for all $i, j, \boldsymbol{W}, q$ with $t \leq T$) are changed to *honest and temporary*, and that a random value $\boldsymbol{s}_{\boldsymbol{x}}[(t, i, j, \boldsymbol{W})]$ is embedded in $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}[\mathsf{sim}^{\mathsf{temp}}]$ for all $i, j, \boldsymbol{W}$. The change is implemented as follows:

– The positions rand and tb_{τ} of $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$ are copied to the positions $\mathsf{rand}^{\mathsf{temp}}$ and $\mathsf{tb}_{\tau}^{\mathsf{temp}}$ respectively, and then the positions rand and tb_{τ} are set to 0.

Table 10 The hybrid $\widetilde{\mathsf{H}}_3$ followed by the loop hybrids and $\widetilde{\mathsf{H}}_4$ in the security proof of 1-UAWS for the case where the ciphertext challenge comes after the secret key query

Hybrid	Vector	rand, tb_τ	$\mathsf{rand}^{\mathsf{temp}}$, $\mathsf{tb}_\tau^{\mathsf{temp}}$	sim	$\mathsf{sim}^{\mathsf{temp}}$
$\widetilde{\mathsf{H}}_3$	$\boldsymbol{v}_{k,q}$	$\boxed{0, 0}$	$0, 0$	$s_{k,f}[q]$	0
	$\boldsymbol{u}_{k,t\le T,i,j,\boldsymbol{W}}$	$0, 0$	$0, 0$	$s_{\boldsymbol{x}}[(t, i, j, \boldsymbol{W})]$	0
		rand, acc	$\mathsf{rand}^{\mathsf{temp}}$, $\mathsf{acc}^{\mathsf{temp}}$	sim	$\mathsf{sim}^{\mathsf{temp}}$
	$\widetilde{\boldsymbol{v}}_{k,q}$	Normal	$\boxed{\text{Normal}}$	0	0
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	$\boxed{0, 0}$	$\boxed{\text{Normal}}$	$s_{\boldsymbol{x}}[(T+1, i, j, \boldsymbol{W})]$	0
		rand, acc	$\mathsf{rand}^{\mathsf{temp}}$, $\mathsf{acc}^{\mathsf{temp}}$	sim	$\mathsf{sim}^{\mathsf{temp}}$
$\widetilde{\mathsf{H}}_{3,q,1}$	$\widetilde{\boldsymbol{v}}_{k,q'<q}$	Normal	$0, 0$	$s_{k,f}[q']$	0
	$\widetilde{\boldsymbol{v}}_{k,q}$	normal	Normal	0	0
	$\widetilde{\boldsymbol{v}}_{k,q'>q}$	Normal	Normal	0	0
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	$0, 0$	Normal	$s_{\boldsymbol{x}}[(T+1, i, j, \boldsymbol{W})]$	0
$\widetilde{\mathsf{H}}_{3,q,2}$	$\widetilde{\boldsymbol{v}}_{k,q'<q}$	Normal	$0, 0$	$s_{k,f}[q']$	0
	$\widetilde{\boldsymbol{v}}_{k,q}$	$\boxed{0, 0}$	$\boxed{0, 0}$	0	1
	$\widetilde{\boldsymbol{v}}_{k,q'>q}$	Normal	normal	0	0
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	$0, 0$	Normal	$s_{\boldsymbol{x}}[(T+1, i, j, \boldsymbol{W})]$	$\boxed{\text{honest } \ell_{k,T+1,i,j,\boldsymbol{W},q} = -\boldsymbol{r}_{\boldsymbol{x}}[(T, i, j, \boldsymbol{W})]\boldsymbol{r}_{k,f}[q] + \cdots}$

Table 10 continued

Hybrid	Vector	rand, tb_τ	$\mathsf{rand}^{\mathsf{temp}}$, $\mathsf{tb}_\tau^{\mathsf{temp}}$	sim	$\mathsf{sim}^{\mathsf{temp}}$
$\widetilde{\mathsf{H}}_{3,q,3}$	$\widetilde{\boldsymbol{v}}_{k,q'<q}$	Normal	0, 0	$s_{k,f}[q']$	0
	$\widetilde{\boldsymbol{v}}_{k,q}$	0, 0	0, 0	0	1
	$\widetilde{\boldsymbol{v}}_{k,q'>q}$	Normal	Normal	0	0
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	0, 0	Normal	$s_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$	$\boxed{\ell_{k,T+1,i,j,\boldsymbol{W},q} \xleftarrow{\$} \mathbb{Z}_p}$
$\widetilde{\mathsf{H}}_{3,q,4}$	$\widetilde{\boldsymbol{v}}_{k,q'<q}$	Normal	0, 0	$s_{k,f}[q']$	0
	$\widetilde{\boldsymbol{v}}_{k,q}$	0, 0	0, 0	0	1
	$\widetilde{\boldsymbol{v}}_{k,q'>q}$	Normal	Normal	0	0
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	0, 0	Normal	$s_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$	$\boxed{\text{simulated } \ell_{k,T+1,i,j,\boldsymbol{W},q} = s_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]s_{k,f}[q]}$
$\widetilde{\mathsf{H}}_{3,q,5} \equiv \widetilde{\mathsf{H}}_{3,q+1,1}$	$\widetilde{\boldsymbol{v}}_{k,q'<q}$	Normal	0, 0	$s_{k,f}[q']$	0
	$\widetilde{\boldsymbol{v}}_{k,q}$	$\boxed{\text{Normal}}$	0, 0	$\boxed{s_{k,f}[q]}$	$\boxed{0}$
	$\widetilde{\boldsymbol{v}}_{k,q'>q}$	Normal	Normal	0	0
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	0, 0	Normal	$s_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$	$\boxed{0}$
Hybrid	Vector	rand, tb_τ	$\mathsf{rand}^{\mathsf{temp}}$, $\mathsf{tb}_\tau^{\mathsf{temp}}$	sim	$\mathsf{sim}^{\mathsf{temp}}$

Table 10 continued

Hybrid	Vector	rand, tb_τ	$\mathsf{rand}^{\mathsf{temp}}$, $\mathsf{tb}_\tau^{\mathsf{temp}}$	sim	$\mathsf{sim}^{\mathsf{temp}}$
$\widetilde{\mathsf{H}}_4$	$\boldsymbol{v}_{k,q}$	$\boxed{\text{Normal}}$	$0, 0$	$\boldsymbol{s}_{k,f}[q]$	0
	$\boldsymbol{u}_{k,t\le T,i,j,\boldsymbol{W}}$	$0, 0$	$0, 0$	$\boldsymbol{s}_{\boldsymbol{x}}[(t, i, j, \boldsymbol{W})]$	0
		rand, acc	$\mathsf{rand}^{\mathsf{temp}}$, $\mathsf{acc}^{\mathsf{temp}}$	sim	$\mathsf{sim}^{\mathsf{temp}}$
	$\widetilde{\boldsymbol{v}}_{k,q}$	Normal	$0, 0$	$\boldsymbol{s}_{k,f}[q]$	0
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	$0, 0$	$\boxed{0, 0}$	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1, i, j, \boldsymbol{W})]$	0

Table 11 The notes of Table 10

For brevity, $\boldsymbol{u}_{\mathsf{init}}, \boldsymbol{v}_{\mathsf{init}}, \boldsymbol{v}_{k,q}, \boldsymbol{u}_{k,t\le T,i,j,\boldsymbol{W}}$ are suppressed.

The reversely sampled $\ell_{k,\mathsf{init}}$ is hardwired in $\boldsymbol{u}_{k,\mathsf{init}}$, and is only needed (and can only be computed so by the reduction) in the exponent of G_1:

$$[\![\ell_{k,\mathsf{init}}]\!]_1 \leftarrow \mathsf{RevSamp}((M_k, 1^N, 1^T, 1^{2^S}), \boldsymbol{x}, [\![\boldsymbol{z}[k]M_k(\boldsymbol{x}) + \beta_k]\!]_1, ([\![\ell_{k,t,\theta_k}]\!]_1)_{t\in[T+1],\theta_k\in\mathcal{C}_{M_k,N,S}}).$$

The omitted term "$\cdots$": $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}[\mathsf{sim}^{\mathsf{temp}}] = \boldsymbol{y}_k[q]\boldsymbol{z}[k]$.

The "normal" in	$\mathsf{rand}, \mathsf{rand}^{\mathsf{temp}}$	$\mathsf{acc}, \mathsf{acc}^{\mathsf{temp}}$
$\widetilde{\boldsymbol{v}}_{k,q}$:	$-\boldsymbol{r}_{k,f}[q]$	$\boldsymbol{y}_k[q]$
$\boldsymbol{u}_{k,T+1,i,j,\boldsymbol{W}}$:	$\boldsymbol{r}_{\boldsymbol{x}}[(T,i,j,\boldsymbol{W})]$	$\boldsymbol{z}[k]$

In this iteration, the labels $\ell_{k,T+1,i,j,\boldsymbol{W},q'}$ with q' are computed as:

	$q' < q$	$q' = q$	$q' > q$
$\widetilde{\mathsf{H}}_3$:	HT	HT	HT
$\widetilde{\mathsf{H}}_{3,q,1\sim5}$:	S	$\boxed{\text{HT} \to \text{HW} \to \text{RW} \to \text{SW} \to \text{S}}$	HT
$\widetilde{\mathsf{H}}_4$:	S	S	S

The shorthands are **H**onest, **R**andom, **S**imulated, hard**W**ired, **T**emporary.

The net effect is that $\ell_{k,T+1,i,j,\boldsymbol{W},q}$'s change from *honest and temporary* to *simulated.*

Note that, in this iteration, $\ell_{k,t\le T,i,j,\boldsymbol{W},q}$'s are unchanged for all q and are already simulated.

The hybrid $\widetilde{\mathsf{H}}_3$ starts after the outer loop of Table 6 finishes, i.e. after the hybrid $\mathsf{H}_{3,T,5}$ and the hybrid $\widetilde{\mathsf{H}}_4$ is identical to the hybrid H_4 (c.f. Table 3).

- The value at $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}[\mathsf{sim}^{\mathsf{temp}}]$ is set to $\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]$. It sets the stage for the inner loop which will make the label values $\ell_{k,t,i,j,\boldsymbol{W},q}$'s as *simulated and temporary*.
- The positions rand and tb_τ of $\boldsymbol{v}_{k,q}$ are copied to the positions $\mathsf{rand}^{\mathsf{temp}}$ and $\mathsf{tb}_\tau^{\mathsf{temp}}$ respectively.

As one can verify that the inner products between the vectors are unchanged, the indistinguishability between the hybrids $\mathsf{H}_{3,t,1}$ and $\mathsf{H}_{3,t,2}$ is guaranteed by the function hiding security of IPFE.

Hybrid $\mathsf{H}_{3,t,4}$. It proceeds identical to $\widetilde{\mathsf{H}}_{3,t,2}$ except that the modes of $\ell_{k,t,i,j,\boldsymbol{W},q}$'s (for all $i, j, \boldsymbol{W}, q$ with $t \le T$) are changed from *honest and temporary* to *simulated and temporary*. This is implemented by $\boldsymbol{v}_{k,q}$'s have their values cleared at $\mathsf{rand}^{\mathsf{temp}}$, $\mathsf{tb}_\tau^{\mathsf{temp}}$, and $\boldsymbol{v}_{k,q}[\mathsf{sim}^{\mathsf{temp}}]$ is set to $\boldsymbol{s}_{k,f}[q]$. We show that $\mathsf{H}_{3,t,2} \approx \mathsf{H}_{3,t,4}$ by a sequence of hybrids used by the inner loop.

Hybrid $\mathsf{H}_{3,t,5}$. It proceeds identical to $\widetilde{\mathsf{H}}_{3,t,4}$ except that the modes of $\ell_{k,t,i,j,\boldsymbol{W},q}$'s (for all $i, j, \boldsymbol{W}, q$ with $t \le T$) are changed from *simulated and temporary* to *simulated*. Moreover, some clean-up work is done in preparation of the next iteration. The change is implemented as follows:

- The positions $\mathsf{rand}^{\mathsf{temp}}$, $\mathsf{tb}_\tau^{\mathsf{temp}}$ and $\mathsf{sim}^{\mathsf{temp}}$ of $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$ are set to 0.
- The value at $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}[\mathsf{sim}]$ is changed from 0 to $\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]$.
- The positions $\mathsf{sim}^{\mathsf{temp}}$ of $\boldsymbol{v}_{k,q}$ is set to 0.

Since the inner products between the vectors $\boldsymbol{u}$'s and $\boldsymbol{v}$'s are unchanged, the indistinguishability between the hybrids $\mathsf{H}_{3,t,4}$ and $\mathsf{H}_{3,t,4}$ is ensured by the function hiding security of IPFE. We observe that $\mathsf{H}_{3,1,1} \equiv \mathsf{H}_2$ and $\mathsf{H}_{3,t,5} \equiv \mathsf{H}_{3,t+1,1}$.

Now, we discuss the hybrids of the inner loop running over $q = 1, \ldots, Q$, which switches the mode of $\ell_{k,t,i,j,\boldsymbol{W},q}$ from *honest and temporary* to *simulated and temporary*.

Hybrid $\mathsf{H}_{3,t,3,q,1}$. It proceeds identical to $\mathsf{H}_{3,t,2}$, except that for $q' < q$, all the $\boldsymbol{v}_{k,q'}$ have their values at $\mathsf{rand}^{\mathsf{temp}}$, $\mathsf{tb}_\tau^{\mathsf{temp}}$'s cleared, and the value $\boldsymbol{s}_{k,f}[q']$ is embedded at $\boldsymbol{v}_{k,q'}[\mathsf{sim}^{\mathsf{temp}}]$. This means that the labels $\ell_{k,t,i,j,\boldsymbol{W},q'}$ for all $i, j, \boldsymbol{W}$ with $t \le T$ and $q' < q$ have been changed from *honest and temporary* to *simulated and temporary*.

Hybrid $\mathsf{H}_{3,t,3,q,2}$. It proceeds identical to $\mathsf{H}_{3,t,3,q,1}$ except that all occurrence of $\boldsymbol{r}_{k,f}[q]$ and $\boldsymbol{s}_{k,f}[q]$ are moved from $\boldsymbol{v}_{k,q'}$'s to $\boldsymbol{u}_{k,t',i,j,\boldsymbol{W},q}$'s using the compensation identity (Notes of Tables 4, 8), for all $q' \neq q$. Further, to make $\widetilde{\boldsymbol{v}}_{k,q}$ free of $\boldsymbol{r}_{k,f}[q]$, it's positions rand, acc are set to zero and $\mathsf{sim}^{\mathsf{temp}}$ is set to 1, and the labels $\ell_{k,T+1,i,j,\boldsymbol{W},q}$'s are hardwired at $\mathsf{sim}^{\mathsf{temp}}$ of $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$ (hence they are in *honest and hardwired* mode). Thus, the labels with $q' = q$ or $(T \ge)t' > t$ or $q' > q$ are computed using the compensation identity on top of their existing mode, and the labels $\ell_{k,t,i,j,\boldsymbol{W},q}$ for all $i, j, \boldsymbol{W}$ become *honest and hardwired* (more specifically, hardwired in $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}[\mathsf{sim}^{\mathsf{comp}}]$). The inner products between $\boldsymbol{u}, \widetilde{\boldsymbol{u}}$'s and $\boldsymbol{v}, \widetilde{\boldsymbol{v}}$'s are unchanged due to these modifications. Hence, the indistinguishability between the hybrids $\mathsf{H}_{3,t,3,q,1}$ and $\mathsf{H}_{3,t,3,q,2}$ follows from the function hiding security of IPFE.

Hybrid $\mathsf{H}_{3,t,3,q,3}$. It proceeds identical to $\mathsf{H}_{3,t,3,q,2}$ except the labels $\ell_{k,t,i,j,\boldsymbol{W},q}$ (for all $i, j, \boldsymbol{W}$ with $t \le T$) hardwired in $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}[\mathsf{sim}^{\mathsf{comp}}]$ become *random and hardwired*. The hybrids $\mathsf{H}_{3,t,3,q,2}$ and $\mathsf{H}_{3,t,3,q,3}$ are indistinguishable by the DDH assumption in $\mathbb{G}_1$.

Hybrid $\mathsf{H}_{3,t,3,q,4}$. It proceeds identical to $\mathsf{H}_{3,t,3,q,3}$ except the labels $\ell_{k,t,i,j,\boldsymbol{W},q}$ (for all $i, j, \boldsymbol{W}$ with $t \le T$) hardwired in $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}[\mathsf{sim}^{\mathsf{comp}}]$ become *simulated and hardwired*, i.e. $\ell_{k,t,i,j,\boldsymbol{W},q} = \boldsymbol{s}_{\boldsymbol{x}}[(t, i, j, \boldsymbol{W})]\boldsymbol{s}_{k,f}[q]$. The hybrids $\mathsf{H}_{3,t,3,q,3}$ and $\mathsf{H}_{3,t,3,q,4}$ are again indistinguishable by the DDH assumption in $\mathbb{G}_1$.

Hybrid $\mathsf{H}_{3,t,3,q,5}$. It proceeds identical to $\mathsf{H}_{3,t,3,q,4}$ except that all occurrences of $\boldsymbol{r}_{k,f}[q]$ and $\boldsymbol{s}_{k,f}[q]$ are moved back to $\boldsymbol{v}_{k,q}$'s, and in the second IPFE, all the vectors are restored back to their initial form, i.e. $\boldsymbol{r}_{k,f}[q]$ is moved back to $\widetilde{\boldsymbol{v}}_{k,q}$. Further, some clean-up work is done in order to prepare the vectors for the next iteration. The values at the position $\mathsf{sim}^{\mathsf{comp}}$ of the vectors $\boldsymbol{v}_{k,q}$ and $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$ are cleared, which means that the labels lose their *compensation* mode and the labels $\ell_{k,t,i,j,\boldsymbol{W},q}$ (for all $i, j, \boldsymbol{W}$ with $t \le T$) become *simulated and temporary*. Also, the values at the position $\mathsf{sim}^{\mathsf{temp}}$ of $\widetilde{\boldsymbol{v}}_{k,q}$ and $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$ are cleared, which in turn ensures that the labels $\ell_{k,T+1,i,j,\boldsymbol{W},q}$'s are changed from *honest hardwired* to *honest* mode. It is easy to see that inner products between $\boldsymbol{u}, \widetilde{\boldsymbol{u}}$'s and $\boldsymbol{v}, \widetilde{\boldsymbol{v}}$'s are unchanged, and hence the indistinguishability between the hybrids $\mathsf{H}_{3,t,3,q,4}$ and $\mathsf{H}_{3,t,3,q,5}$ follows from the function hiding security of IPFE. We observe that $\mathsf{H}_{3,t,3,q,5} \equiv \mathsf{H}_{3,t,3,q+1,1}$, and hence $\mathsf{H}_{3,t,2} \approx \mathsf{H}_{3,t,4}$ in the outer loop hybrids of Table 6.

Note that, the two-level loop ends with the hybrid $\mathsf{H}_{3,T,5}$ where the labels $\ell_{k,t,i,j,\boldsymbol{W},q}$ for all $t \le T$ and for all $i, j, \boldsymbol{W}$ are *simulated*. We now go to the *part 2* of the proof.

Part 2 The sequence of hybrids in the second loop running over $q = 1, \ldots, Q$ (for simulating the labels associated to $t = T + 1$) with two additional hybrids and their indistinguishability arguments.

Hybrid $\widetilde{\mathsf{H}}_3$. It is identical to $\mathsf{H}_{3,T,5}$ except the positions rand, tb_τ of $\boldsymbol{v}_{k,q}$ are set to zero (in the *first* IPFE), and the positions rand, acc of the vectors $\widetilde{\boldsymbol{v}}_{k,q}$'s and $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$'s are copied to their counterparts with superscript temp. Moreover, the positions rand, acc of $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$'s are cleared, which means that the labels $\ell_{k,T+1,i,j,\boldsymbol{W},q}$'s are in *honest and temporary* mode. The inner products between $\boldsymbol{u}, \widetilde{\boldsymbol{u}}$'s and $\boldsymbol{v}, \widetilde{\boldsymbol{v}}$'s are unchanged, and hence the indistinguishability between the hybrids $\mathsf{H}_{3,T,5}$ and $\widetilde{\mathsf{H}}_3$ is guaranteed by the function hiding security of IPFE.

Hybrid $\widetilde{\mathsf{H}}_{3,q,1}$. It proceeds identical to $\widetilde{\mathsf{H}}_3$ except that for $q' < q$, all the $\widetilde{\boldsymbol{v}}_{k,q'}$ have their values at $\mathsf{rand}^{\mathsf{temp}}$, $\mathsf{acc}^{\mathsf{temp}}$'s cleared, and the value $\boldsymbol{s}_{k,f}[q']$ is embedded at $\widetilde{\boldsymbol{v}}_{k,q'}[\mathsf{sim}]$. This means that the labels $\ell_{k,T+1,i,j,\boldsymbol{W},q'}$ for all $i, j, \boldsymbol{W}$ and $q' < q$ have been changed from *honest and temporary* to *simulated*.

Hybrid $\widetilde{\mathsf{H}}_{3,q,2}$. It proceeds identical to $\widetilde{\mathsf{H}}_{3,q,1}$ except that the positions rand, acc, $\mathsf{rand}^{\mathsf{temp}}$, $\mathsf{acc}^{\mathsf{temp}}$ of $\widetilde{\boldsymbol{v}}_{k,q}$ are cleared and $\widetilde{\boldsymbol{v}}_{k,q}[\mathsf{sim}^{\mathsf{temp}}]$ is set to 1. Further, the labels $\ell_{k,T+1,i,j,\boldsymbol{W},q}$ (for all $i, j, \boldsymbol{W}$) are hardwired at $\mathsf{sim}^{\mathsf{temp}}$ of $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$, which means the labels are in *honest and hardwired* mode. The inner products between $\widetilde{\boldsymbol{u}}$'s and $\widetilde{\boldsymbol{v}}$'s are unchanged due to these modifications. Hence, the indistinguishability between the hybrids $\widetilde{\mathsf{H}}_{3,q,1}$ and $\widetilde{\mathsf{H}}_{3,q,2}$ follows from the function hiding security of IPFE.

Hybrid $\widetilde{\mathsf{H}}_{3,q,3}$. It proceeds identical to $\widetilde{\mathsf{H}}_{3,q,2}$ except the labels $\ell_{k,T+1,i,j,\boldsymbol{W},q}$ (for all $i, j, \boldsymbol{W}$) hardwired in $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}[\mathsf{sim}^{\mathsf{temp}}]$ become *random and hardwired*. The hybrids $\widetilde{\mathsf{H}}_{3,q,2}$ and $\widetilde{\mathsf{H}}_{3,q,3}$ are indistinguishable by the DDH assumption in $\mathbb{G}_1$.

Hybrid $\widetilde{\mathsf{H}}_{3,q,4}$. It proceeds identical to $\widetilde{\mathsf{H}}_{3,q,3}$ except the labels $\ell_{k,T+1,i,j,\boldsymbol{W},q}$ (for all $i, j, \boldsymbol{W}$) hardwired in $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}[\mathsf{sim}^{\mathsf{temp}}]$ become *simulated and hardwired*, i.e. $\ell_{k,T+1,i,j,\boldsymbol{W},q} = \boldsymbol{s}_{\boldsymbol{x}}[(T+1, i, j, \boldsymbol{W})]\boldsymbol{s}_{k,f}[q]$. The hybrids $\widetilde{\mathsf{H}}_{3,q,3}$ and $\widetilde{\mathsf{H}}_{3,q,4}$ are again indistinguishable by the DDH assumption in $\mathbb{G}_1$.

Hybrid $\widetilde{\mathsf{H}}_{3,q,5}$. It proceeds identical to $\widetilde{\mathsf{H}}_{3,q,4}$ except that all occurrences of $\boldsymbol{r}_{k,f}[q]$ and $\boldsymbol{s}_{k,f}[q]$ are moved back to $\widetilde{\boldsymbol{v}}_{k,q}$'s, and some clean-up work is done in order to prepare the vectors for the next iteration. The values at the position $\mathsf{sim}^{\mathsf{temp}}$ of the vectors $\widetilde{\boldsymbol{v}}_{k,q}$ and $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$ are cleared, which means that the labels $\ell_{k,T+1,i,j,\boldsymbol{W},q}$ (for all $i, j, \boldsymbol{W}$) become *simulated*. It is easy to see that inner products between $\widetilde{\boldsymbol{u}}$'s and $\widetilde{\boldsymbol{v}}$'s are unchanged, and hence the indistinguishability between the hybrids $\widetilde{\mathsf{H}}_{3,q,4}$ and $\widetilde{\mathsf{H}}_{3,q,5}$ follows from the function hiding security of IPFE. We observe that $\widetilde{\mathsf{H}}_{3,q,5} \equiv \widetilde{\mathsf{H}}_{3,q+1,1}$.

Hybrid $\widetilde{\mathsf{H}}_4$. It is identical to $\widetilde{\mathsf{H}}_{3,Q,5}$ except $\boldsymbol{r}_{k,f}[q]$'s are put back to $\boldsymbol{v}_{k,q}$'s and the positions $\mathsf{rand}^{\mathsf{temp}}$, $\mathsf{acc}^{\mathsf{temp}}$ of $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$ are set to zero. The inner products between $\boldsymbol{u}, \widetilde{\boldsymbol{u}}$'s and $\boldsymbol{v}, \widetilde{\boldsymbol{v}}$'s are unchanged, and hence the indistinguishability between the hybrids $\widetilde{\mathsf{H}}_{3,Q,5}$ and $\widetilde{\mathsf{H}}_4$ is guaranteed by the function hiding security of IPFE.

Lastly, we note that $\mathsf{H}_{3,1,1} \equiv \mathsf{H}_2$ and $\widetilde{\mathsf{H}}_4 \equiv \mathsf{H}_4$ (cf. Table 3). Therefore, $\mathsf{H}_2 \approx \mathsf{H}_4$ in the case of SK before CT. □

6 1-Slot FE for unbounded AWS for L

In this section, we construct a *public key* 1-slot FE scheme for the *unbounded* attribute-weighted sum functionality for L. The scheme satisfies the same properties as of the $\mathsf{SK\text{-}UAWS}^{\mathsf{L}}_{(1,1,1)}$. However, the *public key* scheme supports releasing polynomially many secret keys and a single challenge ciphertext, hence we denote the scheme as $\mathsf{PK\text{-}UAWS}^{\mathsf{L}}_{(\mathsf{poly},1,1)}$.

Along with the AKGS for Logspace Turing machines we require a *function-hiding slotted* $\mathsf{IPFE} = (\mathsf{IPFE.Setup}, \mathsf{IPFE.KeyGen}, \mathsf{IPFE.Enc}, \mathsf{IPFE.SlotEnc}, \mathsf{IPFE.Dec})$ based on G, where $\mathsf{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_{\mathrm{T}}, g_1, g_2, e)$ is pairing group tuple of prime order p.

6.1 The construction

We now describe the $\mathsf{PK\text{-}UAWS}^{\mathsf{L}}_{(\mathsf{poly},1,1)} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$.

$\mathsf{Setup}(1^\lambda)$: On input the security parameter, fix a prime integer $p \in \mathbb{N}$ and define the slots for generating two pair of IPFE master keys as follows:

$$\begin{aligned}
\mathcal{S}_{\mathsf{pub}} &= \left\{\mathsf{index}_1, \mathsf{index}_2, \mathsf{pad}, \mathsf{init}^{\mathsf{pub}}, \mathsf{rand}^{\mathsf{pub}}, \mathsf{acc}^{\mathsf{pub}}\right\} \cup \{\mathsf{tb}_\tau^{\mathsf{pub}} | \tau \in \mathcal{T}\},\\
\mathcal{S}_{\mathsf{copy}} &= \left\{\mathsf{init}^{\mathsf{copy}}, \mathsf{rand}^{\mathsf{copy}}\right\} \cup \{\mathsf{tb}_\tau^{\mathsf{copy}} | \tau \in \mathcal{T}\},\\
\mathcal{S}_{\mathsf{priv}} &= \mathcal{S}_{\mathsf{copy}} \cup \mathcal{S}_{1\text{-}\mathsf{UAWS}} \cup \{\mathsf{pad}^{\mathsf{copy}}, \mathsf{pad}^{\mathsf{temp}}, \mathsf{acc}^{\mathsf{perm}}, \mathsf{sim}^{\mathsf{copy}}\},\\
\widetilde{\mathcal{S}}_{\mathsf{pub}} &= \{\mathsf{index}_1, \mathsf{index}_2, \mathsf{rand}^{\mathsf{pub}}, \mathsf{acc}^{\mathsf{pub}}\},\\
\widetilde{\mathcal{S}}_{1,\mathsf{copy}} &= \{\mathsf{rand}_1^{\mathsf{copy}}, \mathsf{acc}_1^{\mathsf{copy}}\}, \widetilde{\mathcal{S}}_{2,\mathsf{copy}} = \{\mathsf{rand}_2^{\mathsf{copy}}, \mathsf{acc}_2^{\mathsf{copy}}\},\\
\widetilde{\mathcal{S}}_{\mathsf{priv}} &= \widetilde{\mathcal{S}}_{1,\mathsf{copy}} \cup \widetilde{\mathcal{S}}_{2,\mathsf{copy}} \cup \widetilde{\mathcal{S}}_{1\text{-}\mathsf{UAWS}} \cup \{\mathsf{sim}^{\mathsf{copy}}\}
\end{aligned}$$

It generates $(\mathsf{IPFE.MPK}, \mathsf{IPFE.MSK}) \leftarrow \mathsf{IPFE.Setup}(\mathcal{S}_{\mathsf{pub}}, \mathcal{S}_{\mathsf{priv}})$ and $(\widetilde{\mathsf{IPFE.MPK}}, \widetilde{\mathsf{IPFE.MSK}}) \leftarrow \mathsf{IPFE.Setup}(\widetilde{\mathcal{S}}_{\mathsf{pub}}, \widetilde{\mathcal{S}}_{\mathsf{priv}})$ and returns $\mathsf{MSK} = (\mathsf{IPFE.MSK}, \widetilde{\mathsf{IPFE.MSK}})$ and $\mathsf{MPK} = (\mathsf{IPFE.MPK}, \widetilde{\mathsf{IPFE.MPK}})$.

$\mathsf{KeyGen}(\mathsf{MSK}, (\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}}))$: On input the master secret key $\mathsf{MSK} = (\mathsf{IPFE.MSK}, \widetilde{\mathsf{IPFE.MSK}})$ and a function tuple $\boldsymbol{M} = (M_k)_{k \in \mathcal{I}_{\boldsymbol{M}}}$ indexed w.r.t. an index set $\mathcal{I}_{\boldsymbol{M}} \subset \mathbb{N}$ of arbitrary size , it parses $M_k = (Q_k, \boldsymbol{y}_k, \delta_k) \in \mathsf{TM}\ \forall k \in \mathcal{I}_{\boldsymbol{M}}$ and samples the set of elements

$$\left\{\alpha, \beta_k \leftarrow \mathbb{Z}_p \mid k \in \mathcal{I}_{\boldsymbol{M}}, \sum_k \beta_k = 0 \mod p\right\}.$$

It computes a secret key $\mathsf{IPFE.SK}_{\mathsf{pad}} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{\mathsf{pad}}]\!]_2)$ for the following vector $\boldsymbol{v}_{\mathsf{pad}}$:

vector	index_1	index_2	pad	$\mathsf{init}^{\mathsf{pub}}$	$\mathsf{rand}^{\mathsf{pub}}$	$\mathsf{acc}^{\mathsf{pub}}$	$\mathsf{tb}_\tau^{\mathsf{pub}}$	in $\mathcal{S}_{\mathsf{priv}}$
$\boldsymbol{v}_{\mathsf{pad}}$	0	0	α	0	0	0	0	0

For all $k \in \mathcal{I}_{\boldsymbol{M}}$, it proceeds as follows:

1. For $M_k = (Q_k, \boldsymbol{y}_k, \delta_k)$, compute transition blocks $\mathbf{M}_{k,\tau} \in \{0,1\}^{Q_k \times Q_k}, \forall \tau \in \mathcal{T}_k$.
2. Sample independent random vector $\boldsymbol{r}_{k,f} \leftarrow \mathbb{Z}_p^{Q_k}$ and a random element $\pi_k \in \mathbb{Z}_p$.
3. For the following vector $\boldsymbol{v}_{k,\mathsf{init}}$, compute a secret key $\mathsf{IPFE.SK}_{k,\mathsf{init}} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{k,\mathsf{init}}]\!]_2)$:

vector	index_1	index_2	pad	$\mathsf{init}^{\mathsf{pub}}$	$\mathsf{rand}^{\mathsf{pub}}$	$\mathsf{acc}^{\mathsf{pub}}$	$\mathsf{tb}_\tau^{\mathsf{pub}}$	in $\mathcal{S}_{\mathsf{priv}}$
$\boldsymbol{v}_{k,\mathsf{init}}$	π_k	$k \cdot \pi_k$	0	$\boldsymbol{r}_{k,f}[1]$	0	β_k	0	0

4. For each $q \in [Q_k]$, compute the following secret keys

$$\mathsf{IPFE.SK}_{k,q} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{k,q}]\!]_2) \quad \text{and}$$
$$\widetilde{\mathsf{IPFE.SK}}_{k,q} \leftarrow \mathsf{IPFE.KeyGen}(\widetilde{\mathsf{IPFE.MSK}}, [\![\widetilde{\boldsymbol{v}}_{k,q}]\!]_2)$$

where the vectors $\boldsymbol{v}_{k,q}, \widetilde{\boldsymbol{v}}_{k,q}$ are defined as follows:

vector	index_1	index_2	pad	$\mathsf{init}^{\mathsf{pub}}$	$\mathsf{rand}^{\mathsf{pub}}$	$\mathsf{acc}^{\mathsf{pub}}$	$\mathsf{tb}_\tau^{\mathsf{pub}}$	in $\mathcal{S}_{\mathsf{priv}}$
$\boldsymbol{v}_{k,q}$	π_k	$k \cdot \pi_k$	0	0	$-\boldsymbol{r}_{k,f}[q]$	0	$(\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q]$	0

vector	index$_1$	index$_2$	rand$^{\mathsf{pub}}$	acc$^{\mathsf{pub}}$	in $\widetilde{\mathcal{S}}_{\mathsf{priv}}$
$\widetilde{\boldsymbol{v}}_{k,q}$	k	$k \cdot \pi_k$	$-\boldsymbol{r}_{k,f}[q]$	$\alpha \cdot \boldsymbol{y}_k[q]$	0

Finally, it returns the secret key as

$$\mathsf{SK}_{(\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}})} = \left((\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}}), \mathsf{IPFE.SK}_{\mathsf{pad}}, \left\{ \mathsf{IPFE.SK}_{k,\mathsf{init}}, \{\mathsf{IPFE.SK}_{k,q}, \widetilde{\mathsf{IPFE.SK}}_{k,q}\}_{q\in[Q_k]} \right\}_{k\in\mathcal{I}_{\boldsymbol{M}}} \right).$$

$\mathsf{Enc}(\mathsf{MPK}, (\boldsymbol{x}, 1^T, 1^{2^S}), \boldsymbol{z})$: On input the master public key $\mathsf{MPK} = (\mathsf{IPFE.MPK}, \widetilde{\mathsf{IPFE.MPK}})$, a public attribute $\boldsymbol{x} \in \{0,1\}^N$ for some arbitrary $N \geq 1$ with time and space complexity bounds given by $T, S \geq 1$ (as $1^T, 1^{2^S}$) respectively, and the private attribute $\boldsymbol{z} \in \mathbb{Z}_p^n$ for some arbitrary $n \geq 1$, it samples $s \leftarrow \mathbb{Z}_p$ and compute a ciphertext $\mathsf{IPFE.CT}_{\mathsf{pad}} \leftarrow \mathsf{IPFE.Enc}(\mathsf{IPFE.MPK}, [\![\boldsymbol{u}_{\mathsf{pad}}]\!]_1)$ for the vector $\boldsymbol{u}_{\mathsf{pad}}$:

vector	index$_1$	index$_2$	pad	init$^{\mathsf{pub}}$	rand$^{\mathsf{pub}}$	acc$^{\mathsf{pub}}$	tb$_\tau^{\mathsf{pub}}$	in $\mathcal{S}_{\mathsf{priv}}$
$\boldsymbol{u}_{\mathsf{pad}}$	0	0	s	0	0	0	0	0

Next, it does the following:

1. Sample a random vector $\boldsymbol{r}_{\boldsymbol{x}} \leftarrow \mathbb{Z}_p^{[0,T]\times[N]\times[S]\times\{0,1\}^S}$.
2. For each $k \in [n]$, do the following:
   (a) Sample a random element $\rho_k \leftarrow \mathbb{Z}_p$.
   (b) Compute a ciphertext $\mathsf{IPFE.CT}_{k,\mathsf{init}} \leftarrow \mathsf{IPFE.SlotEnc}(\mathsf{IPFE.MPK}, [\![\boldsymbol{u}_{k,\mathsf{init}}]\!]_1)$ for the vector $\boldsymbol{u}_{k,\mathsf{init}}$:

vector	index$_1$	index$_2$	pad	init$^{\mathsf{pub}}$	rand$^{\mathsf{pub}}$	acc$^{\mathsf{pub}}$	tb$_\tau^{\mathsf{pub}}$	in $\mathcal{S}_{\mathsf{priv}}$
$\boldsymbol{u}_{k,\mathsf{init}}$	$-k \cdot \rho_k$	ρ_k	0	$s \cdot \boldsymbol{r}_{\boldsymbol{x}}[(0,1,1,\boldsymbol{0}_S)]$	0	s	0	$\perp$

   (c) For all $t \in [T], i \in [N], j \in [S], \boldsymbol{W} \in \{0,1\}^S$, do the following:
      (i) Compute the transition coefficients $c_\tau(\boldsymbol{x}; t, i, j, \boldsymbol{W}; \boldsymbol{r}_{\boldsymbol{x}}), \forall \tau \in \mathcal{T}$ using $\boldsymbol{r}_{\boldsymbol{x}}$.
      (ii) Compute $\mathsf{IPFE.CT}_{k,t,i,j,\boldsymbol{W}} \leftarrow \mathsf{IPFE.SlotEnc}(\mathsf{IPFE.MPK}, [\![\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}]\!]_1)$ for the vector $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$:

vector	index$_1$	index$_2$	pad	init$^{\mathsf{pub}}$	rand$^{\mathsf{pub}}$	acc$^{\mathsf{pub}}$	tb$_\tau^{\mathsf{pub}}$	in $\mathcal{S}_{\mathsf{priv}}$
$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$	$-k \cdot \rho_k$	ρ_k	0	0	$s \cdot \boldsymbol{r}_{\boldsymbol{x}}[(t-1,i,j,\boldsymbol{W})]$	0	$s \cdot c_\tau(\boldsymbol{x}; t, i, j, \boldsymbol{W}; \boldsymbol{r}_{\boldsymbol{x}})$	$\perp$

   (d) For $t = T+1$, and for all $i \in [N], j \in [S], \boldsymbol{W} \in \{0,1\}^S$, compute $\widetilde{\mathsf{IPFE.CT}}_{k,T+1,i,j,\boldsymbol{W}} \leftarrow \mathsf{IPFE.SlotEnc}(\widetilde{\mathsf{IPFE.MPK}}, [\![\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}]\!]_1)$ for the vector $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$:

vector	index$_1$	index$_2$	rand$^{\mathsf{pub}}$	acc$^{\mathsf{pub}}$	in $\widetilde{\mathcal{S}}_{\mathsf{priv}}$
$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	$-k \cdot \rho_k$	ρ_k	$s \cdot \boldsymbol{r}_{\boldsymbol{x}}[(T,i,j,\boldsymbol{W})]$	$s \cdot \boldsymbol{z}[k]$	$\perp$

3. Finally, it returns the ciphertext as

$$\mathsf{CT}_{(\boldsymbol{x},T,S)} = \Bigg((\boldsymbol{x}, T, S), n, \mathsf{IPFE.CT}_{\mathsf{pad}}, \Big\{ \mathsf{IPFE.CT}_{k,\mathsf{init}}, \{\mathsf{IPFE.CT}_{k,t,i,j,\boldsymbol{W}}\}_{t\in[T]},$$
$$\widetilde{\mathsf{IPFE.CT}}_{k,T+1,i,j,\boldsymbol{W}} \Big\}_{k\in[n], i\in[N], j\in[S], \boldsymbol{W}\in\{0,1\}^S} \Bigg).$$

Dec$(\mathsf{SK}_{(\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}})}, \mathsf{CT}_{(\boldsymbol{x},T,S)})$: On input a secret key $\mathsf{SK}_{(\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}})}$ and a ciphertext $\mathsf{CT}_{(\boldsymbol{x},T,S)}$, do the following:

1. Parse $\mathsf{SK}_{(\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}})}$ and $\mathsf{CT}_{(\boldsymbol{x},T,S)}$ as follows:

$$\mathsf{SK}_{(\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}})} = \Bigg(\big((M_k)_{k\in\mathcal{I}_{\boldsymbol{M}}}, \mathcal{I}_{\boldsymbol{M}}\big), \mathsf{IPFE.SK}_{\mathsf{pad}}, \Big\{\mathsf{IPFE.SK}_{k,\mathsf{init}}, \{\mathsf{IPFE.SK}_{k,q}, \widetilde{\mathsf{IPFE.SK}}_{k,q}\}_{q\in[Q_k]}\Big\}_{k\in\mathcal{I}_{\boldsymbol{M}}}\Bigg), M_k = (Q_k, \boldsymbol{y}_k, \delta_k),$$

$$\mathsf{CT}_{(\boldsymbol{x},T,S)} = \Bigg((\boldsymbol{x}, T, S), n, \mathsf{IPFE.CT}_{\mathsf{pad}}, \Big\{\mathsf{IPFE.CT}_{k,\mathsf{init}}, \{\mathsf{IPFE.CT}_{k,t,i,j,\boldsymbol{W}}\}_{t\in[T]}, \widetilde{\mathsf{IPFE.CT}}_{k,T+1,i,j,\boldsymbol{W}}\Big\}_{k\in[n],i\in[N],j\in[S],\boldsymbol{W}\in\{0,1\}^S}\Bigg).$$

2. Output $\perp$, if $\mathcal{I}_{\boldsymbol{M}} \not\subset [n]$. Else, select the sequence of ciphertexts for the indices $k \in \mathcal{I}_{\boldsymbol{M}}$ as

$$\mathsf{CT}_{(\boldsymbol{x},T,S)} = \Bigg((\boldsymbol{x}, T, S), \Big\{\mathsf{IPFE.CT}_{k,\mathsf{init}}, \{\mathsf{IPFE.CT}_{k,t,i,j,\boldsymbol{W}}\}_{t\in[T]}, \widetilde{\mathsf{IPFE.CT}}_{k,T+1,i,j,\boldsymbol{W}}\Big\}_{k\in\mathcal{I}_{\boldsymbol{M}},i\in[N],j\in[S],\boldsymbol{W}\in\{0,1\}^S}\Bigg).$$

3. Use IPFE decryption to obtain $[\![\mu_{\mathsf{pad}}]\!]_{\mathrm{T}} \leftarrow \mathsf{IPFE.Dec}(\mathsf{IPFE.SK}_{\mathsf{pad}}, \mathsf{IPFE.CT}_{\mathsf{pad}})$.
4. Recall that $\forall k \in \mathcal{I}_{\boldsymbol{M}}, \mathcal{C}_{M_k,N,S} = [N] \times [S] \times \{0,1\}^S \times [Q_k]$, and that we denote any element in it as $\theta_k = (i, j, \boldsymbol{W}, q) \in \mathcal{C}_{M_k,N,S}$ where the only component in the tuple θ_k depending on k is $q \in [Q_k]$. Invoke the IPFE decryption to compute all label values as:

$$\begin{aligned}
&\forall k \in \mathcal{I}_{\boldsymbol{M}} : [\![\ell_{k,\mathsf{init}}]\!]_{\mathrm{T}} = \mathsf{IPFE.Dec}(\mathsf{IPFE.SK}_{k,\mathsf{init}}, \mathsf{IPFE.CT}_{k,\mathsf{init}})\\
&\forall k \in \mathcal{I}_{\boldsymbol{M}}, t \in [T], \theta_k = (i, j, \boldsymbol{W}, q) \in \mathcal{C}_{M_k,N,S} :\\
&\qquad [\![\ell_{k,t,\theta_k}]\!]_{\mathrm{T}} = \mathsf{IPFE.Dec}(\mathsf{IPFE.SK}_{k,q}, \mathsf{IPFE.CT}_{k,t,i,j,\boldsymbol{W}})\\
&\forall k \in \mathcal{I}_{\boldsymbol{M}}, \theta_k = (i, j, \boldsymbol{W}, q) \in \mathcal{C}_{M_k,N,S} :\\
&\qquad [\![\ell_{k,T+1,\theta_k}]\!]_{\mathrm{T}} = \mathsf{IPFE.Dec}(\widetilde{\mathsf{IPFE.SK}}_{k,q}, \widetilde{\mathsf{IPFE.CT}}_{k,T+1,i,j,\boldsymbol{W}})
\end{aligned}$$

5. Next, invoke the AKGS evaluation procedure and obtain the combined value

$$[\![\mu]\!]_{\mathrm{T}} = \prod_{k\in\mathcal{I}_{\boldsymbol{M}}} \mathsf{Eval}\left(\left(M_k, 1^N, 1^T, 1^{2^S}, p\right), \boldsymbol{x}, [\![\ell_{k,\mathsf{init}}]\!]_{\mathrm{T}}, \left\{[\![\ell_{k,t,\theta_k}]\!]_{\mathrm{T}}\right\}_{t\in[T+1],\theta_k\in\mathcal{C}_{M_k,N,S}}\right)$$

6. Finally, it returns μ' such that $[\![\mu]\!]_{\mathrm{T}} = ([\![\mu_{\mathsf{pad}}]\!]_{\mathrm{T}})^{\mu'}$, where $g_{\mathrm{T}} = e(g_1, g_2)$. Similar to [8], we assume that the desired attribute-weighted sum lies within a specified polynomial-sized domain so that μ' can be searched via brute-force.

The correctness of our $\mathsf{PK\text{-}UAWS}^{\mathsf{L}}_{(\mathsf{poly},1,1)}$ can be shown similarly to our secret key scheme of the previous section.

Correctness The first step is to observe that all the AKGS label values are correctly computed for the Turing machines M_k with the fixed input $\boldsymbol{x}$. This holds by the correctness of IPFE and AKGS encoding of the iterated matrix-vector product representing any TM computation. The next (and final) correctness follows from the linearity of AKGS.Eval.

First, by the correctness of IPFE, the decryption recovers $[\![\mu_{\mathsf{pad}}]\!]_{\mathrm{T}} = [\![s\alpha]\!]_{\mathrm{T}}$ from $\mathsf{IPFE.SK}_{\mathsf{pad}}$ and $\mathsf{IPFE.CT}_{\mathsf{pad}}$. Next, for all $k \in \mathcal{I}_{\boldsymbol{M}}, \theta_k = (i, j, \boldsymbol{W}, q) \in \mathcal{C}_{M_k,N,S}$, let $L_{k,\mathsf{init}}, L_{k,t,\theta_k}$ be the label functions corresponding to the AKGS garbling of $M_k = (Q_k, \boldsymbol{y}_k, \delta_k)$. By the definitions of vectors $\boldsymbol{v}_{k,\mathsf{init}}, \boldsymbol{u}_{\mathsf{init}}$ and the correctness of IPFE, we have

$$\begin{aligned}\ell_{k,\mathsf{init}} &= (-k\rho_k\pi_k + k\pi_k\rho_k) + s \cdot \boldsymbol{r}_{\boldsymbol{x}}[(0,1,1,\boldsymbol{0}_S)]\boldsymbol{r}_{k,f}[1] + s \cdot \beta_k \\ &= s \cdot (\boldsymbol{r}_0[(1,1,\boldsymbol{0}_S,1)] + \beta_k) \\ &= s \cdot (\boldsymbol{e}^T_{(1,1,\boldsymbol{0}_S,1)}\boldsymbol{r}_0 + \beta_k) = s \cdot L_{k,\mathsf{init}}(\boldsymbol{x}).\end{aligned}$$

Next, $\forall k \in \mathcal{I}_{\boldsymbol{M}}, t \in [T], q \in [Q_k]$, the structures of $\boldsymbol{v}_{k,q}, \boldsymbol{u}_{t,i,j,\boldsymbol{W}}$ and the correctness of IPFE yields

$$\begin{aligned}&\ell_{k,t,i,j,\boldsymbol{W},q} \\ &= (-k\rho_k\pi_k + k\pi_k\rho_k) - s \cdot \boldsymbol{r}_{\boldsymbol{x}}[(t-1,i,j,\boldsymbol{W})]\boldsymbol{r}_{k,f}[q] \\ &\quad + \textstyle\sum_{\tau \in \mathcal{T}} s \cdot c_\tau(\boldsymbol{x}; t,i,j,\boldsymbol{W}; \boldsymbol{r}_{\boldsymbol{x}})(\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f})[q] \\ &= -s \cdot \boldsymbol{r}_{t-1}[(i,j,\boldsymbol{W},q)] + s \cdot \left(\textstyle\sum_{\tau \in \mathcal{T}} c_\tau(\boldsymbol{x}; t,i,j,\boldsymbol{W}; \boldsymbol{r}_{\boldsymbol{x}})\mathbf{M}_{k,\tau}\boldsymbol{r}_{k,f}\right)[q] \\ &= s \cdot L_{k,t,i,j,\boldsymbol{W},q}(\boldsymbol{x})\end{aligned}$$

When $t = T+1$, $\forall k \in \mathcal{I}_{\boldsymbol{M}}, q \in [Q_k]$, the vectors $\widetilde{\boldsymbol{v}}_{k,q}, \widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$ and the $\widetilde{\mathsf{IPFE}}$ correctness again yields

$$\begin{aligned}&\ell_{k,T+1,i,j,\boldsymbol{W},q} \\ &= (-k\rho_k\pi_k + k\pi_k\rho_k) - s \cdot \boldsymbol{r}_{\boldsymbol{x}}[(T,i,j,\boldsymbol{W})]\boldsymbol{r}_{k,f}[q] + \alpha s \cdot \boldsymbol{z}[k]\boldsymbol{y}_k[q] \\ &= -s \cdot (\boldsymbol{r}_T[(i,j,\boldsymbol{W},q)] + \alpha \boldsymbol{z}[k]\left(1_{[N]\times[S]\times\{0,1\}^S} \otimes \boldsymbol{y}_k\right)[(i,j,\boldsymbol{W},q)]) \\ &= s \cdot L_{k,T+1,i,j,\boldsymbol{W},q}(\boldsymbol{x}).\end{aligned}$$

The above label values are computed in the exponent of the target group $\mathbb{G}_{\mathrm{T}}$. Once all these are generated correctly, the linearity of Eval implies that the garbling can be evaluated in the exponent of $\mathbb{G}_{\mathrm{T}}$. Thus, this yields

$$\begin{aligned}[\![\mu]\!]_{\mathrm{T}} &= \prod_{k \in \mathcal{I}_{\boldsymbol{M}}} \mathsf{Eval}\Bigg(\left(M_k, 1^N, 1^T, 1^{2^S}, p\right), \boldsymbol{x}, [\![\ell_{k,\mathsf{init}}]\!]_{\mathrm{T}}, \\ &\quad \left\{[\![\ell_{k,t,\theta_k}]\!]_{\mathrm{T}}\right\}_{t \in [T+1], \theta_k \in \mathcal{C}_{M_k,N,S}}\Bigg) \\ &= [\![\sum_{k \in \mathcal{I}_{\boldsymbol{M}}} \mathsf{Eval}((M_k, 1^N, 1^T, 1^{2^S}, p), \boldsymbol{x}, \ell_{k,\mathsf{init}}, \{\ell_{k,t,\theta_k}\}_{t \in [T+1], \theta_k \in \mathcal{C}_{M_k,N,S}})]\!]_{\mathrm{T}} \\ &= [\![s \cdot \sum_{k \in \mathcal{I}_{\boldsymbol{M}}} (\alpha \boldsymbol{z}[k] \cdot M_k|_{N,T,S}(\boldsymbol{x}) + \beta_k)]\!]_{\mathrm{T}} \\ &= [\![s\alpha \cdot \sum_{k \in \mathcal{I}_{\boldsymbol{M}}} \boldsymbol{z}[k] \cdot M_k|_{N,T,S}(\boldsymbol{x})]\!]_{\mathrm{T}} = [\![s\alpha \cdot \boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{z}]\!]_{\mathrm{T}}\end{aligned}$$

Finally, since $\boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{z}$ is in polynomial range the decryption recovers it by solving the equation $[\![\mu]\!]_{\mathrm{T}} = ([\![\mu_{\mathsf{pad}}]\!]_{\mathrm{T}})^{\mu'}$ for μ' through exhaustive search over the specified range.

6.2 Security analysis

We first describe the simulator of our public key 1-slot UAWS scheme. The Setup^* works exactly the same as honest Setup in the original scheme. Let the simulated master keys are $\mathsf{MSK}^* = (\mathsf{IPFE.MSK}^*, \mathsf{IPFE}.\widetilde{\mathsf{MSK}}^*)$ and $\mathsf{MPK}^* = (\mathsf{IPFE.MPK}^*, \mathsf{IPFE}.\widetilde{\mathsf{MPK}}^*)$. We assume that there are total Φ number of secret key queries and Φ_{pre} be the number of secret keys appears before the challenge ciphertext is computed. Without loss of generality, we assume that the number of states is the same for all the Turing machine in a particular secret key. Let $n_{\max}$ be the maximum length of $\boldsymbol{z}$ allowed to the adversary $\mathcal{A}$. We assume $n_{\max} = \mathsf{poly}\lambda$ as $\mathcal{A}$

is a polynomial time algorithm. The simulator guesses n which is the length of the private attribute z. The remaining algorithms are as follows:

$\mathsf{KeyGen}_0^*(\mathsf{MSK}^*, (\boldsymbol{M}_\phi, \mathcal{I}_{\boldsymbol{M}_\phi}))$: On input the simulated master secret key $\mathsf{MSK}^* = (\mathsf{IPFE.MSK}^*, \widetilde{\mathsf{IPFE.MSK}}^*$ and a function tuple $\boldsymbol{M}_\phi = (M_{\phi,k})_{k \in \mathcal{I}_{\boldsymbol{M}_\phi}}$ indexed w.r.t. an index set $\mathcal{I}_{\boldsymbol{M}_\phi} \subset \mathbb{N}$ of arbitrary size , it parses $M_{\phi,k} = (Q_\phi, \boldsymbol{y}_k, \delta_k) \in \mathsf{TM}\ \forall k \in \mathcal{I}_{\boldsymbol{M}}$ and proceeds as follows:

1. Sample the set of elements
$$\left\{\alpha_\phi, \widehat{\alpha}_\phi, \beta_{\phi,k}, \widehat{\beta}_{\phi,k} \leftarrow \mathbb{Z}_p \mid k \in \mathcal{I}_{\boldsymbol{M}}, \sum_k \beta_{\phi,k} = 0 \mod p, \sum_k \widehat{\beta}_{\phi,k} = 0 \mod p\right\}$$
2. Compute $\mathsf{IPFE.SK}_{\phi,\mathsf{pad}} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{\mathsf{pad}}]\!]_2)$ for the vector $\boldsymbol{v}_{\phi,\mathsf{pad}}$ defined as

vector	pad	$\mathsf{pad}^{\mathsf{copy}}$	other indices
$\boldsymbol{v}_{\mathsf{pad}}$	α_ϕ	$\widehat{\alpha}_\phi$	0

3. For each $k \in \mathcal{I}_{\boldsymbol{M}}$, do the following:
   3.1 For $M_{\phi,k} = (Q_\phi, \boldsymbol{y}_k, \delta_k)$, compute its transition blocks $\mathbf{M}_{\phi,k,\tau} \in \{0,1\}^{Q_\phi \times Q_\phi}, \forall \tau \in \mathcal{T}_k$.
   3.2 Sample independent random vector $\boldsymbol{r}_{\phi,k,f} \leftarrow \mathbb{Z}_p^{Q_\phi}$ and a random element $\pi_k \in \mathbb{Z}_p$.
   3.3 Compute $\mathsf{IPFE.SK}_{\phi,k,\mathsf{init}} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{k,\mathsf{init}}]\!]_2)$ for the vector $\boldsymbol{v}_{\phi,k,\mathsf{init}}$ defined as

vector	index_1	index_2	$\mathsf{init}^{\mathsf{pub}}$	$\mathsf{acc}^{\mathsf{pub}}$	$\mathsf{init}^{\mathsf{copy}}$	$\mathsf{acc}^{\mathsf{copy}}$	other indices
$\boldsymbol{v}_{\phi,k,\mathsf{init}}$	$\pi_{\phi,k}$	$k \cdot \pi_{\phi,k}$	$\boldsymbol{r}_{\phi,k,f}[1]$	$\beta_{\phi,k}$	$\widehat{\boldsymbol{r}}_{\phi,k,f}1$	$\widehat{\beta}_{\phi,k}$	0

   3.4 For each $q \in [Q_\phi]$, compute $\mathsf{IPFE.SK}_{\phi,k,q} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{\phi,k,q}]\!]_2)$ and $\widetilde{\mathsf{IPFE.SK}}_{\phi,k,q} \leftarrow \mathsf{IPFE.KeyGen}(\widetilde{\mathsf{IPFE.MSK}}, [\![\widetilde{\boldsymbol{v}}_{\phi,k,q}]\!]_2)$ where the vectors $\boldsymbol{v}_{\phi,k,q}$, $\widetilde{\boldsymbol{v}}_{\phi,k,q}$ are defined as

vector	index_1	index_2	$\mathsf{rand}^{\mathsf{pub}}$	$\mathsf{tb}_\tau^{\mathsf{pub}}$	$\mathsf{rand}^{\mathsf{copy}}$	$\mathsf{tb}_\tau^{\mathsf{copy}}$	other indices
$\boldsymbol{v}_{\phi,k,q}$	π_k	$k \cdot \pi_{\phi,k}$	$\boldsymbol{r}_{\phi,k,f}[q]$	$(\mathbf{M}_{\phi,k,\tau}\boldsymbol{r}_{\phi,k,f})[q]$	$\widehat{\boldsymbol{r}}_{\phi,k,f}[q]$	$(\mathbf{M}_{\phi,k,\tau}\widehat{\boldsymbol{r}}_{\phi,k,f})[q]$	0

vector	index_1	index_2	$\mathsf{rand}^{\mathsf{pub}}$	$\mathsf{acc}^{\mathsf{pub}}$	$\mathsf{rand}_2^{\mathsf{copy}}$	$\mathsf{acc}_2^{\mathsf{copy}}$	other indices
$\widetilde{\boldsymbol{v}}_{\phi,k,q}$	k	$k \cdot \pi_{\phi,k}$	$-\boldsymbol{r}_{\phi,k,f}[q]$	$\alpha_\phi \cdot \boldsymbol{y}_k[q]$	$-\widehat{\boldsymbol{r}}_{\phi,k,f}[q]$	$\widehat{\alpha}_\phi \cdot \boldsymbol{y}_k[q]$	0

Finally, it returns the secret key as
$$\mathsf{SK}_{(\boldsymbol{M}_\phi,\mathcal{I}_{\boldsymbol{M}_\phi})} = \left((\boldsymbol{M}_\phi, \mathcal{I}_{\boldsymbol{M}_\phi}), \mathsf{IPFE.SK}_{\phi,\mathsf{pad}}, \left\{\mathsf{IPFE.SK}_{\phi,k,\mathsf{init}}, \{\mathsf{IPFE.SK}_{\phi,k,q}, \widetilde{\mathsf{IPFE.SK}}_{\phi,k,q}\}_{q \in [Q_\phi]}\right\}_{k \in \mathcal{I}_{\boldsymbol{M}_\phi}}\right).$$

$\mathsf{Enc}^*(\mathsf{MPK}^*, \mathsf{MSK}^*, (\boldsymbol{x}, 1^T, 1^{2^S}), \mathcal{V}, n)$: On input the master public key $\mathsf{MPK} = (\mathsf{IPFE.MPK}, \widetilde{\mathsf{IPFE.MPK}})$, a public attribute $\boldsymbol{x} \in \{0,1\}^N$ for some arbitrary $N \geq 1$ with time and space complexity bounds given by $T, S \geq 1$ (as $1^T, 1^{2^S}$) respectively, a set $\mathcal{V} = \{(\boldsymbol{M}_\phi, \mathcal{I}_{\boldsymbol{M}_\phi}), \boldsymbol{M}_\phi(\boldsymbol{x})^\top \boldsymbol{z}\}_{\phi \in \Phi_{\mathsf{pre}}}$ and the length of the private arbitrary $n \in \mathbb{N}$, it proceeds as follows:

1. samples $s \leftarrow \mathbb{Z}_p$ and compute a ciphertext $\mathsf{IPFE.CT}_{\mathsf{pad}} \leftarrow \mathsf{IPFE.Enc}(\mathsf{IPFE.MPK}, [\![\boldsymbol{u}_{\mathsf{pad}}]\!]_1)$ for the vector $\boldsymbol{u}_{\mathsf{pad}}$:

vector	in $\mathcal{S}_{\mathsf{pub}}$	$\mathsf{pad}^{\mathsf{copy}}$	other indices
$\boldsymbol{u}_{\mathsf{pad}}$	0	1	0

2. Sample vectors $\boldsymbol{r_x} \leftarrow \mathbb{Z}_p^{[0,T]\times[N]\times[S]\times\{0,1\}^S}$ and $\boldsymbol{s_x} \leftarrow \mathbb{Z}_p^{[T+1]\times[N]\times[S]\times\{0,1\}^S}$.
3. For each $k \in [n]$, do the following:
   (a) Sample a random element $\rho_k \leftarrow \mathbb{Z}_p$.
   (b) Compute a ciphertext $\mathsf{IPFE.CT}_{k,\mathsf{init}} \leftarrow \mathsf{IPFE.SlotEnc}(\mathsf{IPFE.MPK}, [\![\boldsymbol{u}_{k,\mathsf{init}}]\!]_1)$ for the vector $\boldsymbol{u}_{k,\mathsf{init}}$:

vector	index_1	index_2	$\mathsf{init}^{\mathsf{copy}}$	$\mathsf{acc}^{\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$	other indices
$\boldsymbol{u}_{k,\mathsf{init}}$	$-k\cdot\rho_k$	ρ_k	$\boldsymbol{r_x}[(0,1,1,\boldsymbol{0}_S)]$	1	1	0

   (c) For all $t \in [T]$, $i \in [N]$, $j \in [S]$, $\boldsymbol{W} \in \{0,1\}^S$, do the following:
      (i) Compute the transition coefficients $c_\tau(\boldsymbol{x}; t, i, j, \boldsymbol{W}; \boldsymbol{r_x})$, $\forall\tau \in \mathcal{T}$ using $\boldsymbol{r_x}$.
      (ii) Compute the ciphertext $\mathsf{IPFE.CT}_{k,t,i,j,\boldsymbol{W}} \leftarrow \mathsf{IPFE.SlotEnc}(\mathsf{IPFE.MPK}, [\![\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}]\!]_1)$ for the vector $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$:

vector	index_1	index_2	$\mathsf{rand}^{\mathsf{copy}}$	$\mathsf{tb}_\tau^{\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$	other indices
$\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$	$-k\cdot\rho_k$	ρ_k	$\boldsymbol{r_x}[(t-1,i,j,\boldsymbol{W})]$	$c_\tau(\boldsymbol{x}; t, i, j, \boldsymbol{W}; \boldsymbol{r_x})$	$\boldsymbol{s_x}[(t,i,j,\boldsymbol{W})]$	0

   (d) It finds a dummy vector $\boldsymbol{d} \in \mathbb{Z}_p^n$ such that

$$\boldsymbol{M}_\phi(\boldsymbol{x})^\top \boldsymbol{z} = \sum_{k\in\mathcal{I}_{\boldsymbol{M}_\phi}} M_{\phi,k}(\boldsymbol{x})\boldsymbol{z}[k] = \boldsymbol{M}_\phi(\boldsymbol{x})^\top \boldsymbol{d} = \sum_{k\in\mathcal{I}_{\boldsymbol{M}_\phi}} M_{\phi,k}(\boldsymbol{x})\boldsymbol{d}[k]$$

   holds for all $\phi \in [\Phi_{\mathsf{pre}}]$.
   (e) For $t = T+1$, and for all $i \in [N]$, $j \in [S]$, $\boldsymbol{W} \in \{0,1\}^S$, compute the ciphertext $\widetilde{\mathsf{IPFE.CT}}_{k,T+1,i,j,\boldsymbol{W}} \leftarrow \mathsf{IPFE.SlotEnc}(\widetilde{\mathsf{IPFE.MPK}}, [\![\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}]\!]_1)$ for the vector $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$:

vector	index_1	index_2	$\mathsf{rand}_2^{\mathsf{copy}}$	$\mathsf{acc}_2^{\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$	other indices
$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	$-k\cdot\rho_k$	ρ_k	$\boldsymbol{r_x}[(T,i,j,\boldsymbol{W})]$	$\boldsymbol{d}[k]$	$\boldsymbol{s_x}[(T+1,i,j,\boldsymbol{W})]$	0

4. Finally, it returns the ciphertext as

$$\mathsf{CT}_{(\boldsymbol{x},T,S)} = \Bigg((\boldsymbol{x}, T, S), n, \mathsf{IPFE.CT}_{\mathsf{pad}}, \Big\{ \mathsf{IPFE.CT}_{k,\mathsf{init}}, \{\mathsf{IPFE.CT}_{k,t,i,j,\boldsymbol{W}}\}_{t\in[T]}, \widetilde{\mathsf{IPFE.CT}}_{k,T+1,i,j,\boldsymbol{W}} \Big\}_{k\in[n], i\in[N], j\in[S], \boldsymbol{W}\in\{0,1\}^S} \Bigg).$$

$\mathsf{KeyGen}_1^*(\mathsf{MSK}^*, (\boldsymbol{M}_\phi, \mathcal{I}_{\boldsymbol{M}_\phi}, \boldsymbol{M}_\phi(\boldsymbol{x})^\top \boldsymbol{z}))$: On input the simulated master secret key $\mathsf{MSK}^* = (\mathsf{IPFE.MSK}^*, \widetilde{\mathsf{IPFE.MSK}}^*)$ and a function tuple $\boldsymbol{M}_\phi = (M_{\phi,k})_{k\in\mathcal{I}_{\boldsymbol{M}_\phi}}$ indexed w.r.t. an index set $\mathcal{I}_{\boldsymbol{M}_\phi} \subset \mathbb{N}$ of arbitrary size and it's functional value $\boldsymbol{M}_\phi(\boldsymbol{x})^\top \boldsymbol{z}$, it parses $M_{\phi,k} = (Q_\phi, \boldsymbol{y}_k, \delta_k) \in \mathsf{TM}$ $\forall k \in \mathcal{I}_{\boldsymbol{M}}$ and proceeds as follows:

1. Sample the set of elements

$$\left\{\alpha_\phi, \widehat{\alpha}_\phi, \beta_{\phi,k}, \widehat{\beta}_{\phi,k} \leftarrow \mathbb{Z}_p \mid k \in \mathcal{I}_{\boldsymbol{M}}, \sum_k \beta_{\phi,k} = 0 \mod p, \widehat{\beta}_{\phi,k} \text{ satisfies } (*)\right\}$$

where the condition $(*)$ is given by

$$\begin{aligned} \text{if } \mathcal{I}_{\boldsymbol{M}_\phi} \subseteq [n]: &\quad \textstyle\sum_k \widehat{\beta}_{\phi,k} = 0 \mod p \\ \text{if } (\max \mathcal{I}_{\boldsymbol{M}_\phi} > n) \wedge (\min \mathcal{I}_{\boldsymbol{M}_\phi} \leq n): &\quad \widehat{\beta}_{\phi,k} \leftarrow \mathbb{Z}_p \end{aligned}$$

2. Compute $\mathsf{IPFE.SK}_{\phi,\mathsf{pad}} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{\mathsf{pad}}]\!]_2)$ for the vector $\boldsymbol{v}_{\phi,\mathsf{pad}}$ defined as

vector	pad	pad$^{\mathsf{copy}}$	other indices
$\boldsymbol{v}_{\mathsf{pad}}$	α_ϕ	$\widehat{\alpha}_\phi$	0

3. For all $k \in \mathcal{I}_{\boldsymbol{M}}$, do the following:

3.1 For $M_{\phi,k} = (Q_\phi, \boldsymbol{y}_k, \delta_k)$, compute its transition blocks $\mathbf{M}_{\phi,k,\tau} \in \{0,1\}^{Q_\phi \times Q_\phi}$, $\forall \tau \in \mathcal{T}_k$.

3.2 Sample independent random vectors $\boldsymbol{r}_{\phi,k,f}, \boldsymbol{s}_{\phi,k,f} \leftarrow \mathbb{Z}_p^{Q_\phi}$ and a random element $\pi_k \in \mathbb{Z}_p$.

3.3 Compute $\mathsf{IPFE.SK}_{\phi,k,\mathsf{init}} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{k,\mathsf{init}}]\!]_2)$ for the vector $\boldsymbol{v}_{\phi,k,\mathsf{init}}$ defined as

vector	index$_1$	index$_2$	init$^{\mathsf{pub}}$	acc$^{\mathsf{pub}}$	sim$^{\mathsf{copy}}$	other indices
$\boldsymbol{v}_{\phi,k,\mathsf{init}}$	$\pi_{\phi,k}$	$k \cdot \pi_{\phi,k}$	$\boldsymbol{r}_{\phi,k,f}[1]$	$\beta_{\phi,k}$	$\ell_{\phi,k,\mathsf{init}}$	0

3.4 For each $q \in [Q_\phi]$, compute $\mathsf{IPFE.SK}_{\phi,k,q} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{\phi,k,q}]\!]_2)$ and $\widetilde{\mathsf{IPFE.SK}}_{\phi,k,q} \leftarrow \mathsf{IPFE.KeyGen}(\widetilde{\mathsf{IPFE.MSK}}, [\![\widetilde{\boldsymbol{v}}_{\phi,k,q}]\!]_2)$ where the vectors $\boldsymbol{v}_{\phi,k,q}, \widetilde{\boldsymbol{v}}_{\phi,k,q}$ are defined as

vector	index$_1$	index$_2$	rand$^{\mathsf{pub}}$	tb$_\tau^{\mathsf{pub}}$	sim$^{\mathsf{copy}}$	other indices
$\boldsymbol{v}_{\phi,k,q}$	π_k	$k \cdot \pi_{\phi,k}$	$\boldsymbol{r}_{\phi,k,f}[q]$	$(\mathbf{M}_{\phi,k,\tau}\boldsymbol{r}_{\phi,k,f})[q]$	$\boldsymbol{s}_{\phi,k,f}[q]$	0

vector	index$_1$	index$_2$	rand$^{\mathsf{pub}}$	acc$^{\mathsf{pub}}$	sim$^{\mathsf{copy}}$	other indices
$\widetilde{\boldsymbol{v}}_{\phi,k,q}$	k	$k \cdot \pi_{\phi,k}$	$-\boldsymbol{r}_{\phi,k,f}[q]$	$\alpha_\phi \cdot \boldsymbol{y}_k[q]$	$\boldsymbol{s}_{\phi,k,f}[q]$	0

where $\ell_{\phi,k,\mathsf{init}}$ for $\phi > \Phi_{\mathsf{pre}}$ are computed as

$$\ell_{\phi,1,\mathsf{init}} \leftarrow \mathsf{RevSamp}((M_k, 1^N, 1^T, 1^{2^S}), \boldsymbol{x}, \widehat{\alpha}_\phi \boldsymbol{M}_\phi(\boldsymbol{x})^\top \boldsymbol{z} + \widehat{\beta}_{\phi,1}, (\ell_{\phi,k,t,\theta_k})_{t \in [T+1], \theta_k \in \mathcal{C}_{M_k,N,S}})$$

$$\ell_{\phi,k,\mathsf{init}} \leftarrow \mathsf{RevSamp}((M_k, 1^N, 1^T, 1^{2^S}), \boldsymbol{x}, \widehat{\beta}_{\phi,k}, (\ell_{\phi,k,t,\theta_k})_{t \in [T+1], \theta_k \in \mathcal{C}_{M_k,N,S}})$$

and the other label values $(\ell_{k,t,\theta_k})_{t \in [T+1], \theta_k \in \mathcal{C}_{M_k,N,S}}$ are given by $\ell_{k,t,\theta_k} = \boldsymbol{s}_{\boldsymbol{x}}[(t, i, j, \boldsymbol{W})]\boldsymbol{s}_{\phi,k,f}[q]$.

Finally, it returns the secret key as

$$\mathsf{SK}_{(\boldsymbol{M}_\phi, \mathcal{I}_{\boldsymbol{M}_\phi})} = \left((\boldsymbol{M}_\phi, \mathcal{I}_{\boldsymbol{M}_\phi}), \mathsf{IPFE.SK}_{\phi,\mathsf{pad}}, \left\{\mathsf{IPFE.SK}_{\phi,k,\mathsf{init}}, \{\mathsf{IPFE.SK}_{\phi,k,q}, \widetilde{\mathsf{IPFE.SK}}_{\phi,k,q}\}_{q \in [Q_\phi]}\right\}_{k \in \mathcal{I}_{\boldsymbol{M}_\phi}}\right).$$

Theorem 4 *Assuming the* SXDH *assumption holds in* $\mathcal{G}$ *and the* IPFE *is function hiding secure, the above construction of* 1-Slot FE *for* UAWS *is adaptively simulation secure.*

Proof idea We discuss a high level idea of the proof. We use a two-step approach to show the indistinguishability between the real and the ideal world. Let Φ be the total number of secret keys queried by the adversary.

- In the first step, we move everything from the ciphertext vectors from $\mathcal{S}_{\mathsf{pub}}, \widetilde{\mathcal{S}}_{\mathsf{pub}}$ to the private slots $\mathcal{S}_{\mathsf{priv}}, \widetilde{\mathcal{S}}_{\mathsf{priv}}$. Specifically, we use the $\mathcal{S}_{\mathsf{copy}}$ to compute the inner products between the secret key and ciphertext vectors. To enable this computation, the entries of secret key vectors are copied to $\mathcal{S}_{\mathsf{copy}}$. Note that, the slots of $\mathcal{S}_{\mathsf{pub}}, \widetilde{\mathcal{S}}_{\mathsf{pub}}$ of the secret key vectors must be kept as it is as this will facilitate the decryption of adversarially computed ciphertexts.
- The second step is more technically involved and challenging. We go through a loop of Φ iteration similar to the proof technique of [62], however, unlike their work we can not fully randomize the ciphertext since it should lead to a successful decryption by all the queried keys. We crucially apply the *three slot* encryption technique used by [38, 62]. To handle all the pre-ciphertext secret key queries, we first embed a dummy vector into the ciphertext and then restore it to its original form (copied in $\widetilde{\mathcal{S}}_{2,\mathsf{copy}}$) with the dummy vector in place of the challenge (private) attribute. Additionally, we use the private slot $\mathsf{sim}^{\mathsf{copy}}$ to handle the post-ciphertext secret key queries where we embed the functional values directly into the secret keys. In a nutshell, each iteration of the loop takes care of one particular key and uses two independent randomness—$\widehat{\boldsymbol{r}}_{\boldsymbol{x}}$ in $\mathcal{S}_{\mathsf{1\text{-}UAWS}}$, which interacts with that particular key and $\boldsymbol{r}_{\boldsymbol{x}}$ in $\mathcal{S}_{\mathsf{copy}}, \widetilde{\mathcal{S}}_{1,\mathsf{copy}}, \widetilde{\mathcal{S}}_{2,\mathsf{copy}}$, which interacts with all other keys—so that the security of (1-SK, 1-CT, 1-Slot)-FE can be invoked for each key *one-by-one* in the loop.

We now illustrate the formal indistinguishability arguments of all the hybrids in the proof below.

Proof Let $\mathcal{A}$ be a PPT adversary in the security experiment of UAWS. We show that the advantage of $\mathcal{A}$ in distinguishing between the experiments $\mathsf{Expt}^{\mathsf{1\text{-}Slot\text{-}UAWS}}_{\mathcal{A},\mathsf{real}}(1^\lambda)$ and $\mathsf{Expt}^{\mathsf{1\text{-}Slot\text{-}UAWS}}_{\mathcal{A},\mathsf{ideal}}(1^\lambda)$ is negligible by a sequence of hybrid games played between $\mathcal{A}$ and the challenger. Let $((\boldsymbol{x}, 1^T, 1^{2^S}), \boldsymbol{z})$ be the challenge message and $\boldsymbol{z} \in \mathbb{Z}_p^n$. Suppose $\mathcal{A}$ makes Φ number of secret key queries and out of which the first Φ_{pre} are the pre-ciphertext queries. Let n_{max} be the maximum value of n, the length of $\boldsymbol{z}$, i.e., $\mathcal{A}$ can choose the private attribute whose maximum length can be n_{max}. We assume that $\cup_{\phi\in[\Phi]}\mathcal{I}_{\boldsymbol{M}_\phi} \supseteq [n]$, i.e., the union of all the index sets associated to the secret key queries of $\mathcal{A}$ covers the indices of the ciphertext vectors. This is natural to assume since $\mathcal{A}$ would always want to have maximum information about the encoded message.

In the reduction, we use the shorthand "$\propto \boldsymbol{a}$" to indicate that such components are linear in $\boldsymbol{a}$ and efficiently computable given $\boldsymbol{a}$ in the exponent, and that there is only one natural way of computing them. We now proceed to describe the hybrids. □

Hybrid H_0. It is identical to the real experiment $\mathsf{Expt}^{\mathsf{1\text{-}Slot\text{-}UAWS}}_{\mathcal{A},\mathsf{real}}(1^\lambda)$ of 1-Slot−UAWS scheme where the ciphertexts are generated using SlotEnc of IPFE.

Hybrid $\mathsf{H}_{0.1}$. This is exactly the real experiment except the challenger aborts the experiment immediately if the vector length of $\boldsymbol{z}$ is not n', i.e., $n \neq n'$. Suppose $\mathcal{A}$ outputs $\perp$ when the experiment is aborted. Thus, it is easy to see that the advantage of $\mathcal{A}$ in $\mathsf{H}_{0.1}$ is $\frac{1}{n_{\mathsf{max}}}$ times the advantage in H_0. Thus, if the advantage of $\mathcal{A}$ is negligible in H_0, then it is so in $\mathsf{H}_{0.1}$. Hence, in the remaining hybrids we simply write $n' = n$.

Hybrid H_1. It is identical to $\mathsf{H}_{0.1}$ except the vectors of ciphertext are encrypted using normal Enc of IPFE, i.e. using the master secret key and the positions $\boldsymbol{u}|_{\mathcal{S}_{\mathsf{priv}}}, \widetilde{\boldsymbol{u}}|_{\widetilde{\mathcal{S}}_{\mathsf{priv}}}$ of the vectors $\boldsymbol{u}$'s, $\widetilde{\boldsymbol{u}}$'s are changed from $\perp$ to zero. More specifically, all slots of $\mathcal{S}_{\mathsf{priv}}$ for $\boldsymbol{u}_{\mathsf{pad}}, \boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$ and all slots of $\widetilde{\mathcal{S}}_{\mathsf{priv}}$ for $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$ are changed from $\perp$ to zero. The hybrids H_0 and H_1 are indistinguishable by slot-mode correctness of the slotted IPFE.

Hybrid H_2. It is identical to H_1 except the way we compute the inner products between the secret key and ciphertext vectors. Specifically, the ciphertext randomness s is moved to the secret key, and 1 is placed into the ciphertext vectors in the positions of s. We implement this as follows:

- The ciphertext and secret key vector elements are first copied to $\mathsf{pad}^{\mathsf{copy}}$ and the indices $\mathsf{init}^{\mathsf{copy}}, \mathsf{rand}^{\mathsf{copy}}, \mathsf{tb}_{\tau}^{\mathsf{copy}}, \mathsf{acc}^{\mathsf{copy}}$ of $\mathcal{S}_{\mathsf{copy}}$ and $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$.
- Then, the randomness s is shifted from the ciphertext to the secret key vectors. In particular, the position $\mathsf{pad}^{\mathsf{copy}}$ of $\boldsymbol{v}_{\phi,\mathsf{pad}}$ and $\boldsymbol{u}_{\mathsf{pad}}$ are set to $s\alpha_{\phi}$ and 1 respectively. Similarly, the randomness s is moved to all the indices such as $\mathsf{init}^{\mathsf{copy}}, \mathsf{tb}_{\tau}^{\mathsf{copy}}, \mathsf{rand}^{\mathsf{copy}}, \mathsf{acc}^{\mathsf{copy}}$ of the secret key vectors.

The hybrids are depicted in Table 13. Since the inner product between the secret key and ciphertext vectors are unchanged, the indistinguishability between the hybrids H_1 and H_2 follows from the function hiding security of IPFE. This change prepares the secret key randomness to randomized in the next hybrid.

Hybrid H_3. It proceeds identical to H_2 except that the private slots of the secret key vectors are generated with an independent set of randomnesses: random pad $\widehat{\alpha}_{\phi}$, garbling randomness $\widehat{\boldsymbol{r}}_{k,f}[\phi, k, f]$ and random secret shares $\widehat{\beta}_{\phi,k}$ of zero.
The main difference is that in H_2, the randomnesses used in the secret key vectors at $\mathcal{S}_{\mathsf{pub}}$ and $\mathcal{S}_{\mathsf{priv}}$ are the same, but in H_3, the slots of $\mathcal{S}_{\mathsf{pub}}$ and $\mathcal{S}_{\mathsf{priv}}$ are filled with independent sets of randomnesses. We can invoke the DDH assumption in $\mathbb{G}_2$:

$$\underbrace{\{[\![\alpha_{\phi}, \beta_{\phi,k}, \boldsymbol{r}_{\phi,k,f}; s\alpha_{\phi}, s\beta_{\phi,k}, s\boldsymbol{r}_{\phi,k,f}]\!]_2}_{\text{DDH tuple}}\}_{\phi\in[\Phi],k\in\mathcal{I}_{\boldsymbol{M}_{\phi}}}$$

$$\approx \underbrace{\{[\![\alpha_{\phi}, \beta_{\phi,k}, \boldsymbol{r}_{\phi,k,f}; \widehat{\alpha}_{\phi}, \widehat{\beta}_{\phi,k}, \widehat{\boldsymbol{r}}_{\phi,k,f}]\!]_2}_{\text{random tuple}}\}_{\phi\in[\Phi],k\in\mathcal{I}_{\boldsymbol{M}_{\phi}}}$$

If the DDH tuples is used to compute the secret key vectors, then H_2 is simulated, and if the random tuples are used to compute the secret key vectors then H_3 is simulated. Therefore, the indistinguishability between the hybrids H_2 and H_3 is ensured by the DDH assumption in $\mathbb{G}_2$ (Table 12).

Hybrid H_4. It is identical to the hybrid H_3 except we change the ciphertext vectors to prepare for the second step of the loop. More specifically, the changes are implemented using the following steps:

- Sample a random vector $\boldsymbol{s}_{\boldsymbol{x}} \leftarrow \mathbb{Z}_p^{[T+1]\times[N]\times[S]\times\{0,1\}^S}$ and set the $\mathsf{sim}^{\mathsf{copy}}$ position of the vectors $\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$ as $1, \boldsymbol{s}_{\boldsymbol{x}}[(t, i, j, \boldsymbol{W})]$ respectively.
- The position $\mathsf{sim}^{\mathsf{copy}}$ of $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$ is set as $\boldsymbol{s}_{\boldsymbol{x}}[(T+1, i, j, \boldsymbol{W})]$.
- The reduction finds a dummy vector $\boldsymbol{d} \in \mathbb{Z}_p^n$ such that $\boldsymbol{M}_{\phi}(\boldsymbol{x})^{\top}\boldsymbol{z} = \boldsymbol{M}_{\phi}(\boldsymbol{x})^{\top}\boldsymbol{d} = \sum_{k\in[n]} M_{\phi,k}(\boldsymbol{x})\boldsymbol{d}[k]\ \forall\phi \in [\Phi_{\mathsf{pre}}]$.

Table 12 The first few hybrids in the proof of IND-CPA security of our 1-slot UAWS scheme for L

hybrid	vector	pad	$\mathsf{init}^{\mathsf{pub}}, \mathsf{rand}^{\mathsf{pub}}$ $\mathsf{acc}^{\mathsf{pub}}, \mathsf{tb}_T^{\mathsf{pub}}$	$\mathsf{pad}^{\mathsf{copy}}$	in $\mathcal{S}_{\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$
$\mathsf{H}_{0.1}$	$\boldsymbol{v}_{\phi,\mathsf{pad}}$	α_ϕ		0		
	$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$		$\propto(\boldsymbol{r}_{\phi,k,f}, \beta_{\phi,k})$		0	0
	$\boldsymbol{u}_{\mathsf{pad}}$	s		$\perp$		
	$\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$		$\propto(s, s\boldsymbol{r}_{\boldsymbol{x}})$		$\perp$	$\perp$
			$\mathsf{rand}^{\mathsf{pub}}, \mathsf{acc}^{\mathsf{pub}}$	in $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$	in $\widetilde{\mathcal{S}}_{2,\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$
	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$		$\propto(\boldsymbol{r}_{\phi,k,f}, \alpha_\phi)$	0	0	0
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		$\propto(s\boldsymbol{r}_{\boldsymbol{x}}, s\boldsymbol{z})$	$\perp$	$\perp$	$\perp$
H_1	$\boldsymbol{v}_{\phi,\mathsf{pad}}$	α_ϕ		0		
	$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$		$\propto(\boldsymbol{r}_{\phi,k,f}, \beta_{\phi,k})$		0	0
	$\boldsymbol{u}_{\mathsf{pad}}$	s		$\boxed{0}$		
	$\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$		$\propto(s, s\boldsymbol{r}_{\boldsymbol{x}})$		$\boxed{0}$	$\boxed{0}$
			$\mathsf{rand}^{\mathsf{pub}}, \mathsf{acc}^{\mathsf{pub}}$	in $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$	in $\widetilde{\mathcal{S}}_{2,\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$
	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$		$\propto(\boldsymbol{r}_{\phi,k,f}, \alpha_\phi)$	0	0	0
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		$\propto(s\boldsymbol{r}_{\boldsymbol{x}}, s\boldsymbol{z})$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$
H_2	$\boldsymbol{v}_{\phi,\mathsf{pad}}$	α_ϕ		$\boxed{s\alpha_\phi}$		
	$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$		$\propto(\boldsymbol{r}_{\phi,k,f}, \beta_{\phi,k})$		$\boxed{\propto(s\boldsymbol{r}_{\phi,k,f}, s\beta_{\phi,k})}$	0
	$\boldsymbol{u}_{\mathsf{pad}}$	$\boxed{0}$		$\boxed{1}$		
	$\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$		$\boxed{0}$		$\boxed{\propto(1, \boldsymbol{r}_{\boldsymbol{x}})}$	0
			$\mathsf{rand}^{\mathsf{pub}}, \mathsf{acc}^{\mathsf{pub}}$	in $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$	in $\widetilde{\mathcal{S}}_{2,\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$
	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$		$\propto(\boldsymbol{r}_{\phi,k,f}, \alpha_\phi)$	$\boxed{\propto(s\boldsymbol{r}_{\phi,k,f}, \alpha_\phi s)}$	0	0
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		$\boxed{0}$	$\boxed{\propto(\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{z})}$	0	0
H_3	$\boldsymbol{v}_{\phi,\mathsf{pad}}$	α_ϕ		$\boxed{\widehat{\alpha}_\phi}$		
	$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$		$\propto(\boldsymbol{r}_{\phi,k,f}, \beta_{\phi,k})$		$\boxed{\propto(\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\beta}_{\phi,k})}$	0
	$\boldsymbol{u}_{\mathsf{pad}}$	0		1		
	$\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$		0		$\propto(1, \boldsymbol{r}_{\boldsymbol{x}})$	0
			$\mathsf{rand}^{\mathsf{pub}}, \mathsf{acc}^{\mathsf{pub}}$	in $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$	in $\widetilde{\mathcal{S}}_{2,\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$
	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$		$\propto(\boldsymbol{r}_{\phi,k,f}, \alpha_\phi)$	$\boxed{\propto(\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\alpha}_\phi)}$	0	0
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		0	$\propto(\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{z})$	0	0
$\mathsf{H}_4 \equiv \mathsf{H}_{5,1}$	$\boldsymbol{v}_{\phi,\mathsf{pad}}$	α_ϕ		$\widehat{\alpha}_\phi$		
	$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$		$\propto(\boldsymbol{r}_{\phi,k,f}, \beta_{\phi,k})$		$\propto(\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\beta}_{\phi,k})$	0
	$\boldsymbol{u}_{\mathsf{pad}}$	0		1		
	$\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$		0		$\propto(1, \boldsymbol{r}_{\boldsymbol{x}})$	$\boxed{1 \text{ or } \boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]}$
			$\mathsf{rand}^{\mathsf{pub}}, \mathsf{acc}^{\mathsf{pub}}$	in $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$	in $\widetilde{\mathcal{S}}_{2,\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$
	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$		$\propto(\boldsymbol{r}_{\phi,k,f}, \alpha_\phi)$	$\propto(\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\alpha}_\phi)$	0	0
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		0	$\propto(\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{z})$	$\boxed{\propto(\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{d})}$	$\boxed{\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]}$
$\mathsf{H}_{5,1\sim\Phi,1\sim15}$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	

Then, in $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$, all the elements of $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$ are copied to $\widetilde{\mathcal{S}}_{2,\mathsf{copy}}$ with $\boldsymbol{d}$ in place of $\boldsymbol{z}$.

We will change all the pre-ciphertext secret keys (in the second step) in such a way that they only interact with $\widetilde{\mathcal{S}}_{2,\mathsf{copy}}$ of $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$, instead of $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$.

Observe that, the inner products of the vectors $\boldsymbol{u}$'s, $\widetilde{\boldsymbol{u}}$'s with the vectors $\boldsymbol{v}$'s, $\widetilde{\boldsymbol{v}}$'s are unchanged due to these changes because the corresponding positions of $\boldsymbol{v}$'s and $\widetilde{\boldsymbol{v}}$'s are zero. Therefore, the indistinguishability between the hybrids H_3 and H_4 is ensured by the function hiding security of IPFE.

Table 13 The last few hybrids in the proof of IND-CPA security of our 1-slot UAWS scheme for L

hybrid	vector		pad	$\mathsf{init}^{\mathsf{pub}}, \mathsf{rand}^{\mathsf{pub}}$ $\mathsf{acc}^{\mathsf{pub}}, \mathsf{tb}^{\mathsf{pub}}_{\mathcal{T}}$	$\mathsf{pad}^{\mathsf{copy}}$	in $\mathcal{S}_{\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$
$\mathsf{H}_6 \equiv \mathsf{H}_{5,Q,15}$	$\phi \le \Phi_{\mathsf{pre}}$	$\boldsymbol{v}_{\phi,\mathsf{pad}}$	α_ϕ		$\widehat{\alpha}_\phi$		
		$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$		$\propto (\boldsymbol{r}_{\phi,k,f}, \beta_{\phi,k})$		$\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\beta}_{\phi,k})$	0
		$\boldsymbol{u}_{\mathsf{pad}}$	0		1		
		$\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$		0		$\propto (1, \boldsymbol{r}_{\boldsymbol{x}})$	1 or $\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]$
	$\phi > \Phi_{\mathsf{pre}}$	$\boldsymbol{v}_{\phi,\mathsf{pad}}$	α_ϕ		$\widehat{\alpha}_\phi$		
		$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$		$\propto (\boldsymbol{r}_{\phi,k,f}, \beta_{\phi,k})$		$\boxed{0}$	$\boxed{\ell_{\phi,k,\mathsf{init}} \text{ or } \boldsymbol{s}_{\phi,k,f}[q]}$
				$\mathsf{rand}^{\mathsf{pub}}, \mathsf{acc}^{\mathsf{pub}}$	in $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$	in $\widetilde{\mathcal{S}}_{2,\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$
	$\phi \le \Phi_{\mathsf{pre}}$:	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$		$\propto (\boldsymbol{r}_{\phi,k,f}, \alpha_\phi)$	$\boxed{0}$	$\boxed{\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\alpha}_\phi)}$	0
		$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		0	$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{z})$	$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{d})$	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$
	$\phi > \Phi_{\mathsf{pre}}$:	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$		$\propto (\boldsymbol{r}_{\phi,k,f}, \alpha_\phi)$	$\boxed{0}$	0	$\boxed{\boldsymbol{s}_{\phi,k,f}[q]}$
H_7	$\phi \le \Phi_{\mathsf{pre}}$	$\boldsymbol{v}_{\phi,\mathsf{pad}}$	α_ϕ		$\widehat{\alpha}_\phi$		
		$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$		$\propto (\boldsymbol{r}_{\phi,k,f}, \beta_{\phi,k})$		$\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\beta}_{\phi,k})$	0
		$\boldsymbol{u}_{\mathsf{pad}}$	0		1		
		$\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$		0		$\propto (1, \boldsymbol{r}_{\boldsymbol{x}})$	1 or $\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]$
	$\phi > \Phi_{\mathsf{pre}}$	$\boldsymbol{v}_{\phi,\mathsf{pad}}$	α_ϕ		$\widehat{\alpha}_\phi$		
		$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$		$\propto (\boldsymbol{r}_{\phi,k,f}, \beta_{\phi,k})$		0	$\ell_{\phi,k,\mathsf{init}}$ or $\boldsymbol{s}_{\phi,k,f}[q]$
				$\mathsf{rand}^{\mathsf{pub}}, \mathsf{acc}^{\mathsf{pub}}$	in $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$	in $\widetilde{\mathcal{S}}_{2,\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$
	$\phi \le \Phi_{\mathsf{pre}}$:	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$		$\propto (\boldsymbol{r}_{\phi,k,f}, \alpha_\phi)$	0	$\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\alpha}_\phi)$	0
		$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		0	$\boxed{0}$	$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{d})$	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$
	$\phi > \Phi_{\mathsf{pre}}$:	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$		$\propto (\boldsymbol{r}_{\phi,k,f}, \alpha_\phi)$	0	0	$\boldsymbol{s}_{\phi,k,f}[q]$

For brevity, the vectors for computing the labels are not spelled out. The shorthand "$\propto \boldsymbol{a}$" means that the components there are linear in $\boldsymbol{a}$ and efficiently computable given $\boldsymbol{a}$ in the exponent, and that there is only one natural way of computing them (cf. construction of 1-slot UAWS described in the Section 6).
In H_6 and H_7, $\boldsymbol{u}_{k,\mathsf{init}}[\mathsf{sim}^{\mathsf{copy}}] = 1, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}[\mathsf{sim}^{\mathsf{copy}}] = \boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]$ and $\boldsymbol{v}_{\phi,k,\mathsf{init}}[\mathsf{sim}^{\mathsf{copy}}] = \ell_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q} = \boldsymbol{s}_{\phi,k,f}[q]$ where $\ell_{\phi,k,\mathsf{init}}$ for $\phi > \Phi_{\mathsf{pre}}$ are computed as follows:
$\ell_{\phi,1,\mathsf{init}} \leftarrow \mathsf{RevSamp}((M_k, 1^N, 1^T, 1^{2^S}), \boldsymbol{x}, \widehat{\alpha}_\phi \boldsymbol{M}_\phi(\boldsymbol{x})^\top \boldsymbol{z} + \widehat{\beta}_{\phi,1}, (\ell_{k,t,\theta_k})_{t\in[T+1],\theta_k\in\mathcal{C}_{M_k,N,S}})$
$\ell_{\phi,k,\mathsf{init}} \leftarrow \mathsf{RevSamp}((M_k, 1^N, 1^T, 1^{2^S}), \boldsymbol{x}, \widehat{\beta}_{\phi,k}, (\ell_{k,t,\theta_k})_{t\in[T+1],\theta_k\in\mathcal{C}_{M_k,N,S}})$
and the other label values $(\ell_{k,t,\theta_k})_{t\in[T+1],\theta_k\in\mathcal{C}_{M_k,N,S}}$ are all simulated such that $\ell_{k,t,\theta_k} = \boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]\boldsymbol{s}_{\phi,k,f}[q]$.

We have completed the first step of the security analysis. Now, we move toward the second step with the hybrids $\mathsf{H}_{5,1\sim\Phi,1\sim15}$ which is a loop (running over all secret keys) where we handle each secret key in each iteration. Before going to the description of the loop, we present the last hybrid of the loop and the hybrid that is equivalent to the ideal world.

Hybrid H_6. It is identical to H_4 except the pre-ciphertext secret keys now interacts with $\mathcal{S}_{2,\mathsf{copy}}$ and in the post-ciphertext secret keys, the functional values are hardwired. These changes are implemented as follows:

- In the pre-ciphertext secret keys, everything from the positions in $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$ of $\widetilde{\boldsymbol{v}}_{\phi,k,q}$ (for $\phi \in [\Phi_{\mathsf{pre}}]$) are copied to $\widetilde{\mathcal{S}}_{2,\mathsf{copy}}$, and then the positions in $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$ are set zero.
- In the post-ciphertext secret keys, the positions in $\mathcal{S}_{\mathsf{copy}}$ of $\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$ are set to zero, and the positions $\boldsymbol{v}_{\phi,k,\mathsf{init}}[\mathsf{sim}^{\mathsf{copy}}]$ is set as $\ell_{\phi,k,\mathsf{init}}$ and both of $\boldsymbol{v}_{\phi,k,q}[\mathsf{sim}^{\mathsf{copy}}]$, $\widetilde{\boldsymbol{v}}_{\phi,k,q}$ $[\mathsf{sim}^{\mathsf{copy}}]$ are set as $\boldsymbol{s}_{\phi,k,f}[q]$. The label values $\ell_{\phi,k,\mathsf{init}}$'s are computed as follows:

$$\ell_{\phi,1,\mathsf{init}} \leftarrow \mathsf{RevSamp}((M_k, 1^N, 1^T, 1^{2^S}), \boldsymbol{x}, \widehat{\alpha}_\phi \boldsymbol{M}_\phi(\boldsymbol{x})^\top \boldsymbol{z} + \widehat{\beta}_{\phi,1}, (\ell_{\phi,k,t,\theta_k})_{t\in[T+1],\theta_k\in\mathcal{C}_{M_k,N,S}})$$
$$\ell_{\phi,k,\mathsf{init}} \leftarrow \mathsf{RevSamp}((M_k, 1^N, 1^T, 1^{2^S}), \boldsymbol{x}, \widehat{\beta}_{\phi,k}, (\ell_{\phi,k,t,\theta_k})_{t\in[T+1],\theta_k\in\mathcal{C}_{M_k,N,S}})$$

where $\phi > \Phi_{\mathsf{pre}}$ and the other label values $(\ell_{k,t,\theta_k})_{t\in[T+1],\theta_k\in\mathcal{C}_{M_k,N,S}}$ are given by $\ell_{k,t,\theta_k} = \boldsymbol{s_x}[(t,i,j,\boldsymbol{W})]\boldsymbol{s}_{\phi,k,f}[q]$.

Also, the reduction *ignores* the guessing step of all previous hybrids, meaning that it is not required to guess the length of $\boldsymbol{z}$. We show the indistinguishability between the hybrids in the Claim 3 given below.

Hybrid H_7. It is identical to H_6 except it clears the positions in $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$ of $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$. Since the corresponding terms in $\widetilde{\boldsymbol{v}}_{\phi,k,q}$ are already zero, the inner products are unaffected. Therefore, the indistinguishability between the hybrids H_6 and H_7 is guaranteed by the function hiding security of IPFE. We observe that H_7 is the ideal experiment $\mathsf{Expt}^{\mathsf{1\text{-}Slot\text{-}UAWS}}_{\mathcal{A},\mathsf{ideal}}(1^\lambda)$.

The remaining is the proof of the above claim which will complete the proof of the theorem. □

Claim 3 *The hybrids H_4 and H_6 are indistinguishable, i.e., $\mathsf{H}_4 \approx \mathsf{H}_6$.*

Proof We prove the claim through a loop of hybrids $\mathsf{H}_{5,1\sim\Phi,1\sim15}$ running over all secret keys.

Hybrid $\mathsf{H}_{5,\phi,1}$. It is identical to H_4 except the first $\phi-1$ secret keys are modified so that they either interact with the dummy vector $\boldsymbol{d}$ (if they are pre-ciphertext keys) or the functional values are hardwired into them (if they are post-ciphertext keys). In other words, the first $\phi-1$ secret keys are changed as in H_6. The hybrid is shown in Table 14.

Hybrid $\mathsf{H}_{5,\phi,2}$. It is identical to $\mathsf{H}_{5,\phi,1}$ except that a random multiplier $\widehat{s} \leftarrow \mathbb{Z}_p$ is multiplied with the values in $\mathsf{pad}^{\mathsf{copy}}$, $\mathcal{S}_{\mathsf{copy}}$, $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$. Since $\widehat{s}$ is uniform over $\mathbb{Z}_p$, the probability that $\widehat{s} = 0$ is negligible. Therefore, the hybrids $\mathsf{H}_{5,\phi,1}$ and $\mathsf{H}_{5,\phi,2}$ are identically distributed (including the case of $\widehat{s} = 0$).

Hybrid $\mathsf{H}_{5,\phi,3}$. It is identical to $\mathsf{H}_{5,\phi,2}$ except that the inner product between the ϕ-th secret key vectors and the ciphertext vectors are now computed via the slots in $\{\mathsf{pad}^{\mathsf{temp}}\}\cup\mathcal{S}_{\mathsf{1\text{-}UAWS}}$. This change is implemented as follows:

- The position $\mathsf{pad}^{\mathsf{copy}}$ of $\boldsymbol{v}_{\phi,\mathsf{pad}}$ set to zero and $\mathsf{pad}^{\mathsf{temp}}$ is set to $\widehat{\alpha}_\phi$. Also, $\boldsymbol{u}_{\mathsf{pad}}[\mathsf{pad}^{\mathsf{temp}}]$ is set to $\widehat{s}$.
- The positions in $\mathcal{S}_{\mathsf{copy}}$ of the vectors $\boldsymbol{v}_{\phi,k,\mathsf{init}}$, $\boldsymbol{v}_{\phi,k,q}$ are first copied to $\mathcal{S}_{\mathsf{1\text{-}UAWS}}$ without the random multiplier $\widehat{s}$ and then $\mathcal{S}_{\mathsf{copy}}$ is set to zero. Similarly, $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$ of the vectors $\widetilde{\boldsymbol{v}}_{\phi,k,q}$ are copied to $\widetilde{\mathcal{S}}_{\mathsf{1\text{-}UAWS}}$ without the random multiplier $\widehat{s}$ and then $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$ is set to zero.
- The positions $\mathcal{S}_{\mathsf{copy}}$ of the vectors $\boldsymbol{u}_{k,\mathsf{init}}$, $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$ are copied to $\mathcal{S}_{\mathsf{1\text{-}UAWS}}$ and the random multiplier $\widehat{s}$ is multiplied with the newly copied terms. Similarly, the positions $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$ of the vectors $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$ are copied to $\widetilde{\mathcal{S}}_{\mathsf{1\text{-}UAWS}}$ and the random multiplier $\widehat{s}$ is multiplied with the newly copied terms.

We can verify from the Table 15 that the inner products between the vectors are unchanged, hence the indistinguishability between the hybrids holds due to the function hiding security of IPFE.

Hybrid $\mathsf{H}_{5,\phi,4}$. It is identical to $\mathsf{H}_{5,\phi,3}$ except that in the ciphertext vectors, the term $\widehat{s}\boldsymbol{r_x}$ in $\mathcal{S}_{\mathsf{1\text{-}UAWS}}$, $\widetilde{\mathcal{S}}_{\mathsf{1\text{-}UAWS}}$ is replaced by an independent and uniformly chosen random vector $\widehat{\boldsymbol{s}}$. We can invoke the DDH assumption in $\mathbb{G}_1$:

$$\underbrace{[\![\boldsymbol{r_x}, \widehat{s}, \widehat{s}\boldsymbol{r_x}]\!]_1}_{\text{DDH tuple}} \approx \underbrace{[\![\boldsymbol{r_x}, \widehat{s}, \widehat{\boldsymbol{s}}]\!]_1}_{\text{random tuple}} \text{ for } \widehat{\boldsymbol{s}}, \boldsymbol{r_x} \leftarrow \mathbb{Z}_p^{[0,T],\times[N]\times[S]\times\{0,1\}^S}, \quad \widehat{s} \leftarrow \mathbb{Z}_p$$

Table 14 The first two hybrids of the loop $\mathsf{H}_{5,1\sim\phi,1\sim15}$ which continues to the next page...

hybrid	vector		in $\mathcal{S}_{\mathsf{pub}}$	$\mathsf{pad}^{\mathsf{copy}}$	in $\mathcal{S}_{\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$	$\mathsf{pad}^{\mathsf{temp}}$	in $\mathcal{S}_{1\text{-}\mathsf{UAWS}}$
$\mathsf{H}_{5,\phi,1}$	$\phi' < \phi$, $\phi' \le \Phi_{\mathsf{pre}}$	$\boldsymbol{v}_{\phi',\mathsf{pad}}$		$\widehat{\alpha}_{\phi'}$			0	
		$\boldsymbol{v}_{\phi',k,\mathsf{init}}, \boldsymbol{v}_{\phi',k,q}$			$\propto (\widehat{\boldsymbol{r}}_{\phi',k,f}, \widehat{\beta}_{\phi',k})$	0		0
	$\phi' < \phi$, $\phi' > \Phi_{\mathsf{pre}}$	$\boldsymbol{v}_{\phi',\mathsf{pad}}$		$\widehat{\alpha}_{\phi'}$			0	
		$\boldsymbol{v}_{\phi',k,\mathsf{init}}, \boldsymbol{v}_{\phi',k,q}$			0	$\ell_{\phi',k,\mathsf{init}}$ or $\boldsymbol{s}_{\phi',k,f}[q]$		0
		$\boldsymbol{v}_{\phi,\mathsf{pad}}$		$\widehat{\alpha}_{\phi}$			0	
		$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s (independent of $\widehat{\alpha}_\phi, \widehat{\boldsymbol{r}}_{\phi,k,f}$)		$\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\beta}_{\phi,k})$	0		0
		$\boldsymbol{u}_{\mathsf{pad}}$		1			0	
		$\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$			$\propto (1, \boldsymbol{r}_{\boldsymbol{x}})$	1 or $\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]$		0
	$\phi' > \phi$, $\phi' \le \Phi_{\mathsf{pre}}$	$\boldsymbol{v}_{\phi',\mathsf{pad}}$		$\widehat{\alpha}_{\phi'}$			0	
		$\boldsymbol{v}_{\phi',k,\mathsf{init}}, \boldsymbol{v}_{\phi',k,q}$			$\propto (\widehat{\boldsymbol{r}}_{\phi',k,f}, \widehat{\beta}_{\phi',k})$	0		0
	$\phi' > \phi$, $\phi > \Phi_{\mathsf{pre}}$	$\boldsymbol{v}_{\phi',\mathsf{pad}}$		$\widehat{\alpha}_{\phi'}$			0	
		$\boldsymbol{v}_{\phi',k,\mathsf{init}}, \boldsymbol{v}_{\phi',k,q}$			$\propto (\widehat{\boldsymbol{r}}_{\phi',k,f}, \widehat{\beta}_{\phi',k})$	0		0
			in $\widetilde{\mathcal{S}}_{\mathsf{pub}}$	in $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$	in $\widetilde{\mathcal{S}}_{2,\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$		in $\widetilde{\mathcal{S}}_{1\text{-}\mathsf{UAWS}}$
	$\phi' < \phi \le \Phi_{\mathsf{pre}}$:	$\widetilde{\boldsymbol{v}}_{\phi',k,q}$		0	$\propto (\widehat{\boldsymbol{r}}_{\phi',k,f}, \widehat{\alpha}_{\phi'})$	0		0
	$\Phi_{\mathsf{pre}} < \phi' < \phi$:	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$		0	0	$\boldsymbol{s}_{\phi',k,f}[q]$		0
		$\widetilde{\boldsymbol{v}}_{\phi,k,q}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s (independent of $\widehat{\alpha}_\phi, \widehat{\boldsymbol{r}}_{\phi,k,f}$)	$\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\alpha}_{\phi})$	0	0		0
		$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{z})$	$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{d})$	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$		0
	$\Phi_{\mathsf{pre}} \ge \phi' > \phi$:	$\widetilde{\boldsymbol{v}}_{\phi',k,q}$		$\propto (\widehat{\boldsymbol{r}}_{\phi',k,f}, \widehat{\alpha}_{\phi'})$	0	0		0
	$\phi' > \phi > \Phi_{\mathsf{pre}}$:	$\widetilde{\boldsymbol{v}}_{\phi',k,q}$		$\propto (\widehat{\boldsymbol{r}}_{\phi',k,f}, \widehat{\alpha}_{\phi'})$	0	0		0
$\mathsf{H}_{5,\phi,2}$		$\boldsymbol{v}_{\phi,\mathsf{pad}}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	$\boxed{\widehat{s}\widehat{\alpha}_\phi}$			0	
		$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$			$\boxed{\propto (\widehat{s}\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{s}\widehat{\beta}_{\phi,k})}$	0		0
		$\boldsymbol{u}_{\mathsf{pad}}$		1			0	
		$\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$			$\propto (1, \boldsymbol{r}_{\boldsymbol{x}})$	1 or $\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]$		0
			in $\widetilde{\mathcal{S}}_{\mathsf{pub}}$	in $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$	in $\widetilde{\mathcal{S}}_{2,\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$		in $\widetilde{\mathcal{S}}_{1\text{-}\mathsf{UAWS}}$
		$\widetilde{\boldsymbol{v}}_{\phi,k,q}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	$\boxed{\propto (\widehat{s}\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{s}\widehat{\alpha}_{\phi})}$	0	0		0
		$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{z})$	$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{d})$	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$		0

to show the indistinguishability between the hybrids $\mathsf{H}_{5,\phi,3}$ and $\mathsf{H}_{5,\phi,4}$.

Hybrid $\mathsf{H}_{5,\phi,5}$. It is identical to $\mathsf{H}_{5,\phi,4}$ except that in the ciphertext vectors, the term $\widehat{\boldsymbol{s}}$ in $\mathcal{S}_{1\text{-}\mathsf{UAWS}}, \widehat{\mathcal{S}}_{1\text{-}\mathsf{UAWS}}$ is replaced by $\widehat{s}\widehat{\boldsymbol{r}}_{\boldsymbol{x}}$ where we note that $\boldsymbol{r}_{\boldsymbol{x}}$ of $\mathcal{S}_{\mathsf{copy}}$ is independent of this newly sampled $\widehat{\boldsymbol{r}}_{\boldsymbol{x}}$. We invoke the DDH assumption in $\mathbb{G}_1$:

$$\underbrace{[\![\widehat{\boldsymbol{r}}_{\boldsymbol{x}}, \widehat{s}, \widehat{\boldsymbol{s}}]\!]_1}_{\text{random tuple}} \approx \underbrace{[\![\widehat{\boldsymbol{r}}_{\boldsymbol{x}}, \widehat{s}, \widehat{s}\widehat{\boldsymbol{r}}_{\boldsymbol{x}}]\!]_1}_{\mathsf{DDH}\ \text{tuple}} \text{ for } \widehat{\boldsymbol{s}}, \widehat{\boldsymbol{r}}_{\boldsymbol{x}} \leftarrow \mathbb{Z}_p^{[0,T],\times[N]\times[S]\times\{0,1\}^S}, \quad \widehat{s} \leftarrow \mathbb{Z}_p$$

to show the indistinguishability between the hybrids $\mathsf{H}_{5,\phi,4}$ and $\mathsf{H}_{5,\phi,5}$.

Hybrid $\mathsf{H}_{5,\phi,6}$. It is identical to $\mathsf{H}_{5,\phi,5}$ except that the random multiplier $\widehat{s}$ is moved back to the secret key vectors $\boldsymbol{v}_\phi$'s from the ciphertext vectors $\boldsymbol{u}$'s. The indistinguishability between $\mathsf{H}_{5,\phi,6}$ and $\mathsf{H}_{5,\phi,5}$ follows from the function hiding property of IPFE.

Hybrid $\mathsf{H}_{5,\phi,7}$. It is identical to $\mathsf{H}_{5,\phi,6}$ except that the random multiplier $\widehat{s}$ is removed from the secret key vectors. The hybrids $\mathsf{H}_{5,\phi,6}$ and $\mathsf{H}_{5,\phi,7}$ are identically distributed.

Hybrid $\mathsf{H}_{5,\phi,8}$. It is identical to $\mathsf{H}_{5,\phi,7}$ except the ϕ-th secret key (if it is a pre-ciphertext query, i.e. $\phi \in [\Phi_{\mathsf{pre}}]$) now interacts with the dummy vector $\boldsymbol{d}$ or the functional value is hardwired into it (if it is a post-challenge query, i.e. $\phi > \Phi_{\mathsf{pre}}$). This change is implemented as follows:

– If $\phi \in [\Phi_{\mathsf{pre}}]$, then there is no change required in the secret key, but $\boldsymbol{z}$ is replaced by $\boldsymbol{d}$ in the ciphertext vector $\widetilde{\boldsymbol{u}}$'s.

Table 15 The intermediate hybrids $\mathsf{H}_{5,\phi,3}$ to $\mathsf{H}_{5,\phi,7}$ of the loop $\mathsf{H}_{5,1\sim\Phi,1\sim15}$

hybrid	vector	in $\mathcal{S}_{\text{pub}}$	pad^{copy}	in $\mathcal{S}_{\text{copy}}$	sim^{copy}	pad^{temp}	in $\mathcal{S}_{\text{1-UAWS}}$
$\mathsf{H}_{5,\phi,3}$	$\boldsymbol{v}_{\phi,\text{pad}}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	$\boxed{0}$			$\boxed{\widehat{\alpha}_\phi}$	
	$\boldsymbol{v}_{\phi,k,\text{init}}, \boldsymbol{v}_{\phi,k,q}$			$\boxed{0}$	0		$\boxed{\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\beta}_{\phi,k})}$
	$\boldsymbol{u}_{\text{pad}}$		1			$\boxed{\widehat{s}}$	
	$\boldsymbol{u}_{k,\text{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$			$\propto (1, \boldsymbol{r}_{\boldsymbol{x}})$	1 or $\boldsymbol{s}_{\boldsymbol{x}}[(t, i, j, \boldsymbol{W})]$		$\boxed{\propto (\widehat{s}, \widehat{s}\boldsymbol{r}_{\boldsymbol{x}})}$
		in $\widetilde{\mathcal{S}}_{\text{pub}}$	in $\widetilde{\mathcal{S}}_{1,\text{copy}}$	in $\widetilde{\mathcal{S}}_{2,\text{copy}}$	sim^{copy}		in $\widetilde{\mathcal{S}}_{\text{1-UAWS}}$
	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	$\boxed{0}$	0	0		$\boxed{\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\alpha}_\phi)}$
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{z})$	$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{d})$	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1, i, j, \boldsymbol{W})]$		$\boxed{\propto (\widehat{s}\boldsymbol{r}_{\boldsymbol{x}}, \widehat{s}\boldsymbol{z})}$
$\mathsf{H}_{5,\phi,4}$	$\boldsymbol{v}_{\phi,\text{pad}}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	0			$\widehat{\alpha}_\phi$	
	$\boldsymbol{v}_{\phi,k,\text{init}}, \boldsymbol{v}_{\phi,k,q}$			0	0		$\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\beta}_{\phi,k})$
	$\boldsymbol{u}_{\text{pad}}$		1			$\widehat{s}$	
	$\boldsymbol{u}_{k,\text{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$			$\propto (1, \boldsymbol{r}_{\boldsymbol{x}})$	1 or $\boldsymbol{s}_{\boldsymbol{x}}[(t, i, j, \boldsymbol{W})]$		$\boxed{\propto (\widehat{s}, \boldsymbol{s})}$

Table 15 continued

hybrid	vector	in $\mathcal{S}_{\text{pub}}$	pad^{copy}	in $\mathcal{S}_{\text{copy}}$	sim^{copy}	pad^{temp}	in $\mathcal{S}_{\text{1-UAWS}}$
		in $\widetilde{\mathcal{S}}_{\text{pub}}$	in $\widetilde{\mathcal{S}}_{1,\text{copy}}$	in $\widetilde{\mathcal{S}}_{2,\text{copy}}$	sim^{copy}		in $\widetilde{\mathcal{S}}_{\text{1-UAWS}}$
	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	0	0	0		$\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\alpha}_\phi)$
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		$\propto (\boldsymbol{r_x}, \boldsymbol{z})$	$\propto (\boldsymbol{r_x}, \boldsymbol{d})$	$s_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$		$\boxed{\propto (\boldsymbol{s}, \widehat{s\boldsymbol{z}})}$
$\mathsf{H}_{5,\phi,5}$	$\boldsymbol{v}_{\phi,\text{pad}}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	0			$\widehat{\alpha}_\phi$	
	$\boldsymbol{v}_{\phi,k,\text{init}}, \boldsymbol{v}_{\phi,k,q}$			0	0		$\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\beta}_{\phi,k})$
	$\boldsymbol{u}_{\text{pad}}$		1			$\widehat{s}$	
	$\boldsymbol{u}_{k,\text{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$			$\propto (1, \boldsymbol{r_x})$	1 or $\boldsymbol{s_x}[(t,i,j,\boldsymbol{W})]$		$\boxed{\propto (\widehat{s}, \widehat{s\boldsymbol{r}_{\boldsymbol{x}}})}$
		in $\widetilde{\mathcal{S}}_{\text{pub}}$	in $\widetilde{\mathcal{S}}_{1,\text{copy}}$	in $\widetilde{\mathcal{S}}_{2,\text{copy}}$	sim^{copy}		in $\widetilde{\mathcal{S}}_{\text{1-UAWS}}$
	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	0	0	0		$\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\alpha}_\phi)$
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		$\propto (\boldsymbol{r_x}, \boldsymbol{z})$	$\propto (\boldsymbol{r_x}, \boldsymbol{d})$	$s_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$		$\boxed{\propto (\widehat{s\boldsymbol{r}_{\boldsymbol{x}}}, \widehat{s\boldsymbol{z}})}$
$\mathsf{H}_{5,\phi,6}$	$\boldsymbol{v}_{\phi,\text{pad}}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	0			$\boxed{\widehat{s\widehat{\alpha}_\phi}}$	

Table 15 continued

hybrid	vector	in $\mathcal{S}_{\mathsf{pub}}$	$\mathsf{pad}^{\mathsf{copy}}$	in $\mathcal{S}_{\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$	$\mathsf{pad}^{\mathsf{temp}}$	in $\mathcal{S}_{\mathsf{1\text{-}UAWS}}$
	$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$			0	0		$\boxed{\propto (\widehat{s\boldsymbol{r}}_{\phi,k,f}, \widehat{s\beta}_{\phi,k})}$
	$\boldsymbol{u}_{\mathsf{pad}}$		1			$\boxed{1}$	
	$\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$			$\propto (1, \boldsymbol{r}_{\boldsymbol{x}})$	1 or $\boldsymbol{s}_{\boldsymbol{x}}[(t, i, j, \boldsymbol{W})]$		$\boxed{\propto (1, \widehat{\boldsymbol{r}}_{\boldsymbol{x}})}$
		in $\widetilde{\mathcal{S}}_{\mathsf{pub}}$	in $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$	in $\widetilde{\mathcal{S}}_{2,\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$		in $\widetilde{\mathcal{S}}_{\mathsf{1\text{-}UAWS}}$
	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	0	0	0		$\boxed{\propto (\widehat{s\boldsymbol{r}}_{\phi,k,f}, \widehat{s\alpha}_{\phi})}$
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{z})$	$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{d})$	$s_{\boldsymbol{x}}[(T+1, i, j, \boldsymbol{W})]$		$\boxed{\propto (\widehat{\boldsymbol{r}}_{\boldsymbol{x}}, \boldsymbol{z})}$
$\mathsf{H}_{5,\phi,7}$	$\boldsymbol{v}_{\phi,\mathsf{pad}}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	0			$\boxed{\widehat{\alpha}_\phi}$	
	$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$			0	0		$\boxed{\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\beta}_{\phi,k})}$
	$\boldsymbol{u}_{\mathsf{pad}}$		1			1	
	$\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$			$\propto (1, \boldsymbol{r}_{\boldsymbol{x}})$	1 or $\boldsymbol{s}_{\boldsymbol{x}}[(t, i, j, \boldsymbol{W})]$		$\propto (1, \widehat{\boldsymbol{r}}_{\boldsymbol{x}})$
		in $\widetilde{\mathcal{S}}_{\mathsf{pub}}$	in $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$	in $\widetilde{\mathcal{S}}_{2,\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$		in $\widetilde{\mathcal{S}}_{\mathsf{1\text{-}UAWS}}$
	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	0	0	0		$\boxed{\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\alpha}_{\phi})}$
	$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{z})$	$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{d})$	$s_{\boldsymbol{x}}[(T+1, i, j, \boldsymbol{W})]$		$\propto (\widehat{\boldsymbol{r}}_{\boldsymbol{x}}, \boldsymbol{z})$

Table 16 The intermediate hybrids $\mathsf{H}_{5,\phi,8}$ and $\mathsf{H}_{5,\phi,9}$ of the loop $\mathsf{H}_{5,1\sim\phi,1\sim15}$

hybrid	vector		in $\mathcal{S}_{\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$	in $\mathcal{S}_{1\text{-}\mathsf{UAWS}}$: $\mathsf{rand}, \mathsf{acc}, \mathsf{tb}_\tau$	in $\mathcal{S}_{1\text{-}\mathsf{UAWS}}$: sim
$\mathsf{H}_{5,\phi,8}$	$\phi \le \Phi_{\mathsf{pre}}$:	$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$	0	0	$\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\beta}_{\phi,k})$	0
		$\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$	$\propto (1, \boldsymbol{r}_{\boldsymbol{x}})$	1 or $\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]$	$\propto (1, \widehat{\boldsymbol{r}}_{\boldsymbol{x}})$	$\boxed{1 \text{ or } \boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]}$
	$\phi > \Phi_{\mathsf{pre}}$	$\boldsymbol{v}_{\phi,1,\mathsf{init}}$	0	0	$\boxed{0}$	$\boxed{\ell_{\phi,1,\mathsf{init}} \leftarrow \mathsf{RevSamp}(\widehat{\alpha}_\phi \boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{z} + \widehat{\beta}_{\phi,1})}$
		$\boldsymbol{v}_{\phi,k>1,\mathsf{init}}$	0	0	$\boxed{0}$	$\boxed{\ell_{\phi,k,\mathsf{init}} \leftarrow \mathsf{RevSamp}(\widehat{\beta}_{\phi,k})}$
		$\boldsymbol{v}_{\phi,k,q}$	0	0	$\boxed{0}$	$\boxed{\boldsymbol{s}_{\phi,k,f}[q]}$
			in $\widetilde{S}_{2,\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$	in $\mathcal{S}_{1\text{-}\mathsf{UAWS}}$: $\mathsf{rand}, \mathsf{acc}$	in $\mathcal{S}_{1\text{-}\mathsf{UAWS}}$: sim
	$\phi \le \Phi_{\mathsf{pre}}$:	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$	0	0	$\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\alpha}_\phi)$	0
		$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{d})$	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$	$\boxed{\propto (\widehat{\boldsymbol{r}}_{\boldsymbol{x}}, \boldsymbol{d})}$	$\boxed{\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]}$
	$\phi > \Phi_{\mathsf{pre}}$:	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$	0	0	$\boxed{0}$	$\boxed{\boldsymbol{s}_{\phi,k,f}[q]}$
$\mathsf{H}_{5,\phi,9}$	$\phi \le \Phi_{\mathsf{pre}}$:	$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$	0	0	$\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\beta}_{\phi,k})$	0
		$\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$	$\propto (1, \boldsymbol{r}_{\boldsymbol{x}})$	1 or $\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]$	$\propto (1, \widehat{\boldsymbol{r}}_{\boldsymbol{x}})$	$\boxed{0}$
	$\phi > \Phi_{\mathsf{pre}}$	$\boldsymbol{v}_{\phi,1,\mathsf{init}}$	0	$\boxed{\ell_{\phi,1,\mathsf{init}} \leftarrow \mathsf{RevSamp}(\widehat{\alpha}_\phi \boldsymbol{M}(\boldsymbol{x})^\top \boldsymbol{z} + \widehat{\beta}_{\phi,1})}$	0	$\boxed{0}$
		$\boldsymbol{v}_{\phi,k>1,\mathsf{init}}$	0	$\boxed{\ell_{\phi,k,\mathsf{init}} \leftarrow \mathsf{RevSamp}(\widehat{\beta}_{\phi,k})}$	0	$\boxed{0}$
		$\boldsymbol{v}_{\phi,k,q}$	0	$\boxed{\boldsymbol{s}_{\phi,k,f}[q]}$	0	$\boxed{0}$
			in $\widetilde{S}_{2,\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$	in $\mathcal{S}_{1\text{-}\mathsf{UAWS}}$: $\mathsf{rand}, \mathsf{acc}$	in $\mathcal{S}_{1\text{-}\mathsf{UAWS}}$: sim
	$\phi \le \Phi_{\mathsf{pre}}$:	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$	0	0	$\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\alpha}_\phi)$	0
		$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$	$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{d})$	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$	$\propto (\widehat{\boldsymbol{r}}_{\boldsymbol{x}}, \boldsymbol{d})$	$\boxed{0}$
	$\phi > \Phi_{\mathsf{pre}}$:	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$	0	$\boxed{\boldsymbol{s}_{\phi,k,f}[q]}$	0	$\boxed{0}$

- Also, in the ciphertext, the position sim of the vectors $\boldsymbol{u}_{k,\mathsf{init}}$, $\boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$ and $\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$ are set to 1, $\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]$ and $\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$ respectively.
- If $\phi > \Phi_{\mathsf{pre}}$, then everything in $S_{1\text{-}\mathsf{UAWS}}$ and $\widetilde{S}_{1\text{-}\mathsf{UAWS}}$ of the secret key vectors are cleared except the sim position. More specifically, the positions rand, acc, tb_τ of $S_{1\text{-}\mathsf{UAWS}}$ and $\widetilde{S}_{1\text{-}\mathsf{UAWS}}$ are set to zero for $\boldsymbol{v}$'s and $\widetilde{\boldsymbol{v}}$'s, and $\boldsymbol{v}_{\phi,k,\mathsf{init}}[\mathsf{sim}]$ is set as the label values $\ell_{\phi,k,\mathsf{init}}$, and both of $\boldsymbol{v}_{\phi,k,q}[\mathsf{sim}]$, $\widetilde{\boldsymbol{v}}_{\phi,k,q}[\mathsf{sim}]$ are as $\boldsymbol{s}_{\phi,k,f}[q]$.

To make the change as shown in Table 16, we invoke the security of the (1-SK, 1-CT, 1-Slot)-FE scheme. In particular, Theorem 3 is applied for the ϕ-th key and the single challenge ciphertext. Observe that the guessing step is already done in this security proof (i.e., $\mathsf{H}_{0.1}$), hence this step is skipped while we apply the security of (1-SK, 1-CT, 1-Slot)-FE scheme. This makes the reduction more efficient and reduces the security loss incurred due to guessing. Also, we *emphasize* that in this hybrid we utilize the slots index_1 and index_2 of $\mathcal{S}_{1\text{-}\mathsf{UAWS}}$, $\widetilde{S}_{1\text{-}\mathsf{UAWS}}$ through the security reduction of (1-SK, 1-CT, 1-Slot)-FE scheme, which indeed depends on the Lemma 4. Thus, the hybrids $\mathsf{H}_{5,\phi,7}$ and $\mathsf{H}_{5,\phi,8}$ are indistinguishable.

Hybrid $\mathsf{H}_{5,\phi,9}$. It is identical to the hybrid $\mathsf{H}_{5,\phi,8}$ except that everything is copied from the position sim of $S_{1\text{-}\mathsf{UAWS}}$ to the corresponding position $\mathsf{sim}^{\mathsf{copy}}$, and then the position sim is cleared from all $\boldsymbol{u}$'s, $\widetilde{\boldsymbol{u}}$'s and $\boldsymbol{v}_\phi$'s, $\widetilde{\boldsymbol{v}}_\phi$'s. The hybrid is described in Table 16. The purpose of this change is to compute the label values for post-ciphertext secret keys utilizing the position $\mathsf{sim}^{\mathsf{copy}}$ instead of using the slots of $S_{1\text{-}\mathsf{UAWS}}$ and prepare it for handling the next key. Note that, if ϕ-th key is a pre-ciphertext secret key then there no change takes place in $\boldsymbol{v}_\phi$'s and $\widetilde{\boldsymbol{v}}_\phi$'s, however, the sim position of $\boldsymbol{u}$'s and $\widetilde{\boldsymbol{u}}$'s are cleared. We observe that the inner products are unchanged and, hence the indistinguishability between the hybrids $\mathsf{H}_{5,\phi,8}$ and $\mathsf{H}_{5,\phi,9}$ is ensured by the function hiding property of IPFE.

Table 17 The intermediate hybrids $\mathsf{H}_{5,\phi,10}$ to $\mathsf{H}_{5,\phi,13}$ of the loop $\mathsf{H}_{5,1\sim\phi,1\sim15}$

hybrid	vector		in $\mathcal{S}_{\mathsf{pub}}$	$\mathsf{pad}^{\mathsf{copy}}$	in $\mathcal{S}_{\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$	$\mathsf{pad}^{\mathsf{temp}}$	in $\mathcal{S}_{1\text{-}\mathsf{UAWS}}$
$\mathsf{H}_{5,\phi,10}$	$\phi \le \Phi_{\mathsf{pre}}$	$\boldsymbol{v}_{\phi,\mathsf{pad}}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	0			$\boxed{\widehat{s}\widehat{\alpha}_\phi}$	
		$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$			0	0		$\boxed{\propto (\widehat{s}\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{s}\widehat{\beta}_{\phi,k})}$
		$\boldsymbol{u}_{\mathsf{pad}}$		1			1	
		$\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$			$\propto (1, \boldsymbol{r}_{\boldsymbol{x}})$	1 or $\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]$		$\propto (1, \widehat{\boldsymbol{r}}_{\boldsymbol{x}})$
	$\phi > \Phi_{\mathsf{pre}}$	$\boldsymbol{v}_{\phi,\mathsf{pad}}$		$\boxed{\widehat{\alpha}_\phi}$			$\boxed{0}$	
		$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$			0	$\ell_{\phi,k,\mathsf{init}}$ or $\boldsymbol{s}_{\phi,k,f}[q]$		0
			in $\widetilde{\mathcal{S}}_{\mathsf{pub}}$	in $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$	in $\widetilde{\mathcal{S}}_{2,\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$		in $\widetilde{\mathcal{S}}_{1\text{-}\mathsf{UAWS}}$
	$\phi \le \Phi_{\mathsf{pre}}$:	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	0	0	0		$\boxed{\propto (\widehat{s}\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{s}\widehat{\alpha}_\phi)}$
		$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{z})$	$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{d})$	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$		$\propto (\widehat{\boldsymbol{r}}_{\boldsymbol{x}}, \boldsymbol{d})$
	$\phi > \Phi_{\mathsf{pre}}$:	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	0	0	$\boldsymbol{s}_{\phi,k,f}[q]$		0
$\mathsf{H}_{5,\phi,11}$	$\phi \le \Phi_{\mathsf{pre}}$	$\boldsymbol{v}_{\phi,\mathsf{pad}}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	0			$\boxed{\widehat{\alpha}_\phi}$	
		$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$			0	0		$\boxed{\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\beta}_{\phi,k})}$
		$\boldsymbol{u}_{\mathsf{pad}}$		1			$\boxed{\widehat{s}}$	
		$\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$			$\propto (1, \boldsymbol{r}_{\boldsymbol{x}})$	1 or $\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]$		$\boxed{\propto (\widehat{s}, \widehat{s}\widehat{\boldsymbol{r}}_{\boldsymbol{x}})}$
			in $\widetilde{\mathcal{S}}_{\mathsf{pub}}$	in $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$	in $\widetilde{\mathcal{S}}_{2,\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$		in $\widetilde{\mathcal{S}}_{1\text{-}\mathsf{UAWS}}$
	$\phi \le \Phi_{\mathsf{pre}}$:	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	0	0	0		$\boxed{\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\alpha}_\phi)}$
		$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{z})$	$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{d})$	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$		$\boxed{\propto (\widehat{s}\widehat{\boldsymbol{r}}_{\boldsymbol{x}}, \widehat{s}\boldsymbol{d})}$
$\mathsf{H}_{5,\phi,12}$	$\phi \le \Phi_{\mathsf{pre}}$	$\boldsymbol{v}_{\phi,\mathsf{pad}}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	0			$\widehat{\alpha}_\phi$	
		$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$			0	0		$\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\beta}_{\phi,k})$
		$\boldsymbol{u}_{\mathsf{pad}}$		1			$\widehat{s}$	
		$\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$			$\propto (1, \boldsymbol{r}_{\boldsymbol{x}})$	1 or $\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]$		$\boxed{\propto (\widehat{s}, \widehat{\boldsymbol{s}})}$
			in $\widetilde{\mathcal{S}}_{\mathsf{pub}}$	in $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$	in $\widetilde{\mathcal{S}}_{2,\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$		in $\widetilde{\mathcal{S}}_{1\text{-}\mathsf{UAWS}}$
	$\phi \le \Phi_{\mathsf{pre}}$:	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	0	0	0		$\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\alpha}_\phi)$
		$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{z})$	$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{d})$	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$		$\boxed{\propto (\widehat{\boldsymbol{s}}, \widehat{s}\boldsymbol{d})}$
$\mathsf{H}_{5,\phi,13}$	$\phi \le \Phi_{\mathsf{pre}}$	$\boldsymbol{v}_{\phi,\mathsf{pad}}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	0			$\widehat{\alpha}_\phi$	
		$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$			0	0		$\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\beta}_{\phi,k})$
		$\boldsymbol{u}_{\mathsf{pad}}$		1			$\widehat{s}$	
		$\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$			$\propto (1, \boldsymbol{r}_{\boldsymbol{x}})$	1 or $\boldsymbol{s}_{\boldsymbol{x}}[(t,i,j,\boldsymbol{W})]$		$\boxed{\propto (\widehat{s}, \widehat{s}\widehat{\boldsymbol{r}}_{\boldsymbol{x}})}$
			in $\widetilde{\mathcal{S}}_{\mathsf{pub}}$	in $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$	in $\widetilde{\mathcal{S}}_{2,\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$		in $\widetilde{\mathcal{S}}_{1\text{-}\mathsf{UAWS}}$
	$\phi \le \Phi_{\mathsf{pre}}$:	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	0	0	0		$\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\alpha}_\phi)$
		$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{z})$	$\propto (\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{d})$	$\boldsymbol{s}_{\boldsymbol{x}}[(T+1,i,j,\boldsymbol{W})]$		$\boxed{\propto (\widehat{s}\widehat{\boldsymbol{r}}_{\boldsymbol{x}}, \widehat{s}\boldsymbol{d})}$

Hybrid $\mathsf{H}_{5,\phi,10}$. It is identical to $\mathsf{H}_{5,\phi,9}$ except that a random element $\widehat{s} \leftarrow \mathbb{Z}_p$ is multiplied to the secret key vectors $\boldsymbol{v}_\phi$'s and $\widetilde{\boldsymbol{v}}_\phi$'s if $\phi \le \Phi_{\mathsf{pre}}$, i.e. the ϕ-th key under consideration is a pre-challenge secret key. On the other hand, if $\phi > \Phi_{\mathsf{pre}}$ then the position $\mathsf{pad}^{\mathsf{temp}}$ of $\boldsymbol{v}_{\phi,\mathsf{pad}}$ is first copied to $\mathsf{pad}^{\mathsf{copy}}$ and then $\mathsf{pad}^{\mathsf{temp}}$ is cleared. Since $\widehat{s}$ is uniform over $\mathbb{Z}_p$, the probability that $\widehat{s} = 0$ is negligible. The hybrid is described in Table 17. Therefore, the hybrids $\mathsf{H}_{5,\phi,9}$ and $\mathsf{H}_{5,\phi,10}$ are identically distributed (including the case of $\widehat{s} = 0$) if $\phi \le \Phi_{\mathsf{pre}}$. On the other hand, if $\phi > \Phi_{\mathsf{pre}}$ then the hybrids are indistinguishable due to function security of IPFE.

Hybrid $\mathsf{H}_{5,\phi,11}$. It is identical to $\mathsf{H}_{5,\phi,10}$ except that the random multiplier $\widehat{s}$ is moved to the ciphertext vectors $\boldsymbol{u}$'s, $\widetilde{\boldsymbol{u}}$'s from the secret key vectors $\boldsymbol{v}_\phi$'s, $\widetilde{\boldsymbol{v}}_\phi$'s. The indistinguishability between $\mathsf{H}_{5,\phi,10}$ and $\mathsf{H}_{5,\phi,11}$ follows from the function hiding property of IPFE.

Table 18 The final two hybrids $\mathsf{H}_{5,\phi,14}$ and $\mathsf{H}_{5,\phi,15}$ of the loop $\mathsf{H}_{5,1\sim\phi,1\sim15}$

hybrid	vector		in $\mathcal{S}_{\mathsf{pub}}$	$\mathsf{pad}^{\mathsf{copy}}$	in $\mathcal{S}_{\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$	$\mathsf{pad}^{\mathsf{temp}}$	in $\mathcal{S}_{1\text{-}\mathsf{UAWS}}$
$\mathsf{H}_{5,\phi,14}$	$\phi \le \Phi_{\mathsf{pre}}$	$\boldsymbol{v}_{\phi,\mathsf{pad}}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	$\boxed{\widehat{s}\widehat{\alpha}_\phi}$			$\boxed{0}$	
		$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$			$\boxed{\propto (\widehat{s}\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{s}\widehat{\beta}_{\phi,k})}$	0		$\boxed{0}$
		$\boldsymbol{u}_{\mathsf{pad}}$		1			$\boxed{0}$	
		$\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$			$\propto (1, \boldsymbol{r_x})$	1 or $\boldsymbol{s_x}[(t,i,j,\boldsymbol{W})]$		$\boxed{0}$
			in $\widetilde{\mathcal{S}}_{\mathsf{pub}}$	in $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$	in $\widetilde{\mathcal{S}}_{2,\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$		in $\widetilde{\mathcal{S}}_{1\text{-}\mathsf{UAWS}}$
	$\phi \le \Phi_{\mathsf{pre}}$:	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	0	$\boxed{\propto (\widehat{s}\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{s}\widehat{\alpha}_\phi)}$	0		$\boxed{0}$
		$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		$\propto (\boldsymbol{r_x}, \boldsymbol{z})$	$\propto (\boldsymbol{r_x}, \boldsymbol{d})$	$\boldsymbol{s_x}[(T+1,i,j,\boldsymbol{W})]$		$\boxed{0}$
$\mathsf{H}_{5,\phi,15}$ $\equiv$ $\mathsf{H}_{5,\phi+1,1}$	$\phi \le \Phi_{\mathsf{pre}}$	$\boldsymbol{v}_{\phi,\mathsf{pad}}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	$\boxed{\widehat{\alpha}_\phi}$			0	
		$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$			$\boxed{\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\beta}_{\phi,k})}$	0		0
		$\boldsymbol{u}_{\mathsf{pad}}$		1			0	
		$\boldsymbol{u}_{k,\mathsf{init}}, \boldsymbol{u}_{k,t,i,j,\boldsymbol{W}}$			$\propto (1, \boldsymbol{r_x})$	1 or $\boldsymbol{s_x}[(t,i,j,\boldsymbol{W})]$		0
	$\phi > \Phi_{\mathsf{pre}}$	$\boldsymbol{v}_{\phi,\mathsf{pad}}$		$\widehat{\alpha}_\phi$			0	
		$\boldsymbol{v}_{\phi,k,\mathsf{init}}, \boldsymbol{v}_{\phi,k,q}$			0	$\ell_{\phi,k,\mathsf{init}}$ or $\boldsymbol{s}_{\phi,k,f}[q]$		0
			in $\widetilde{\mathcal{S}}_{\mathsf{pub}}$	in $\widetilde{\mathcal{S}}_{1,\mathsf{copy}}$	in $\widetilde{\mathcal{S}}_{2,\mathsf{copy}}$	$\mathsf{sim}^{\mathsf{copy}}$		in $\widetilde{\mathcal{S}}_{1\text{-}\mathsf{UAWS}}$
	$\phi \le \Phi_{\mathsf{pre}}$:	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	0	$\boxed{\propto (\widehat{\boldsymbol{r}}_{\phi,k,f}, \widehat{\alpha}_\phi)}$	0		0
		$\widetilde{\boldsymbol{u}}_{k,T+1,i,j,\boldsymbol{W}}$		$\propto (\boldsymbol{r_x}, \boldsymbol{z})$	$\propto (\boldsymbol{r_x}, \boldsymbol{d})$	$\boldsymbol{s_x}[(T+1,i,j,\boldsymbol{W})]$		0
	$\phi > \Phi_{\mathsf{pre}}$:	$\widetilde{\boldsymbol{v}}_{\phi,k,q}$	$\alpha_\phi, \boldsymbol{r}_{\phi,k,f}$'s	0	0	$\boldsymbol{s}_{\phi,k,f}[q]$		0

Hybrid $\mathsf{H}_{5,\phi,12}$. It is identical to $\mathsf{H}_{5,\phi,11}$ except that in the ciphertext vectors, the term $\widehat{s}\widehat{\boldsymbol{r}}_{\boldsymbol{x}}$ in $\mathcal{S}_{1\text{-}\mathsf{UAWS}}$, $\widetilde{\mathcal{S}}_{1\text{-}\mathsf{UAWS}}$ is replaced by an independent and uniformly chosen random vector $\widehat{\boldsymbol{s}}$. We can invoke the DDH assumption in $\mathbb{G}_1$:

$$\underbrace{[\![\widehat{\boldsymbol{r}}_{\boldsymbol{x}}, \widehat{s}, \widehat{s}\widehat{\boldsymbol{r}}_{\boldsymbol{x}}]\!]_1}_{\text{DDH tuple}} \approx \underbrace{[\![\widehat{\boldsymbol{r}}_{\boldsymbol{x}}, \widehat{s}, \widehat{\boldsymbol{s}}]\!]_1}_{\text{random tuple}} \text{ for } \widehat{\boldsymbol{s}}, \widehat{\boldsymbol{r}}_{\boldsymbol{x}} \leftarrow \mathbb{Z}_p^{[0,T],\times[N]\times[S]\times\{0,1\}^S}, \quad \widehat{s} \leftarrow \mathbb{Z}_p$$

to show the indistinguishability between $\mathsf{H}_{5,\phi,11}$ and $\mathsf{H}_{5,\phi,12}$.

Hybrid $\mathsf{H}_{5,\phi,13}$. It is identical to $\mathsf{H}_{5,\phi,12}$ except that in the ciphertext vectors, the term $\widehat{\boldsymbol{s}}$ in $\mathcal{S}_{1\text{-}\mathsf{UAWS}}$, $\widetilde{\mathcal{S}}_{1\text{-}\mathsf{UAWS}}$ is replaced by $\widehat{s}\boldsymbol{r}_{\boldsymbol{x}}$ where we note that the $\boldsymbol{r}_{\boldsymbol{x}}$ is the same as that of used in the other slots such as $\mathcal{S}_{\mathsf{copy}}$. We invoke the DDH assumption in $\mathbb{G}_1$:

$$\underbrace{[\![\boldsymbol{r}_{\boldsymbol{x}}, \widehat{s}, \widehat{\boldsymbol{s}}]\!]_1}_{\text{random tuple}} \approx \underbrace{[\![\boldsymbol{r}_{\boldsymbol{x}}, \widehat{s}, \widehat{s}\boldsymbol{r}_{\boldsymbol{x}}]\!]_1}_{\text{DDH tuple}} \text{ for } \widehat{\boldsymbol{s}}, \boldsymbol{r}_{\boldsymbol{x}} \leftarrow \mathbb{Z}_p^{[0,T],\times[N]\times[S]\times\{0,1\}^S}, \quad \widehat{s} \leftarrow \mathbb{Z}_p$$

to show the indistinguishability between the hybrids $\mathsf{H}_{5,\phi,12}$ and $\mathsf{H}_{5,\phi,13}$.

Hybrid $\mathsf{H}_{5,\phi,14}$. It is identical to $\mathsf{H}_{5,\phi,13}$ except that the inner product between the ϕ-th secret key vectors and the ciphertext vectors are now computed via the slots in $\{\mathsf{pad}^{\mathsf{copy}}\} \cup \mathcal{S}_{\mathsf{copy}} \cup \widetilde{\mathcal{S}}_{2,\mathsf{copy}}$. This change is implemented as follows:

– The random multiplier $\widehat{s}$ is moved back to the secret key vectors, i.e. $\boldsymbol{v}_\phi$'s and $\widetilde{\boldsymbol{v}}_\phi$'s. The positions in $\mathcal{S}_{1\text{-}\mathsf{UAWS}}$ of the vectors $\boldsymbol{v}_{\phi,k,\mathsf{init}}$, $\boldsymbol{v}_{\phi,k,q}$ are first copied to $\mathcal{S}_{\mathsf{copy}}$, and then $\mathcal{S}_{1\text{-}\mathsf{UAWS}}$ is set to zero. Similarly, the positions in $\widetilde{\mathcal{S}}_{1\text{-}\mathsf{UAWS}}$ of the vectors $\widetilde{\boldsymbol{v}}_{\phi,k,q}$ are first copied to $\widetilde{\mathcal{S}}_{2,\mathsf{copy}}$, and then $\widetilde{\mathcal{S}}_{1\text{-}\mathsf{UAWS}}$ is set to zero.

– The position $\mathsf{pad}^{\mathsf{temp}}$ of $\boldsymbol{v}_{\phi,\mathsf{pad}}$ is copied to $\mathsf{pad}^{\mathsf{copy}}$, and then $\mathsf{pad}^{\mathsf{temp}}$ is cleared.
– The positions $\mathsf{pad}^{\mathsf{temp}}$, $S_{1\text{-}\mathsf{UAWS}}$ and $\widetilde{S}_{1\text{-}\mathsf{UAWS}}$ of the ciphertext vectors $\boldsymbol{u}$'s and $\widetilde{\boldsymbol{u}}$'s are cleared.

We can verify from the Table 18 that the inner products between the vectors are unchanged, hence the indistinguishability between the hybrids holds due to the function hiding security of IPFE.

Hybrid $\mathsf{H}_{5,\phi,15}$. It is identical to $\mathsf{H}_{5,\phi,14}$ except that the random multiplier $\widehat{s}$ is removed from the secret key vectors. The hybrids $\mathsf{H}_{5,\phi,6}$ and $\mathsf{H}_{5,\phi,7}$ are identically distributed.

We observe that $\mathsf{H}_{5,\phi,15} \approx \mathsf{H}_{5,\phi+1,1}$. Also, the guessing of the length of z is not required from the hybrid $\mathsf{H}_{5,\Phi_{\mathsf{pre}}+1,15}$. This is because the reduction knows the length of z while simulating all the post-challenge secret keys. Thus, $\mathsf{H}_{5,\Phi,15} \equiv \mathsf{H}_6$. Therefore, by a hybrid argument we can show that $\mathsf{H}_4 \equiv \mathsf{H}_{5,1,15} \approx \mathsf{H}_{5,\Phi,15} \equiv \mathsf{H}_6$. This completes the proof of the claim. □

7 FE for UAWS for DFA/NFA

In this section, we present the construction of FE for UAWS for deterministic finite automata (DFA). We know that a DFA can be viewed as a Turing machine with space complexity 1 and time complexity N which is the input length. Thus, our FE for UAWS for DFA is a special case of the UAWS for L and NL. We first describe the AKGS construction for DFA from [62] and then present a simplified construction of FE for UAWS for DFA.

Definition 11 A *deterministic finite automata* is a tuple $(Q, \boldsymbol{y}_{\mathsf{acc}}, \delta)$, where $Q \geq 1$ is the number of states (we use [Q] as the set of states and 1 the initial state), $\boldsymbol{y}_{\mathsf{acc}} \in \{0,1\}^Q$ indicates whether each state is accepting, and δ is a (state transition) function between $[Q] \times \{0,1\}$ and $[Q]$. For $\boldsymbol{x} \in \{0,1\}^N$ for some $N \geq 1$, the DFA accepts $\boldsymbol{x}$ if there exits $q_0, \ldots, q_N \in [Q]$ (called an accepting path) such that

$$q_0 = 1, \quad ((q_{i-1}, \boldsymbol{x}[i]), q_i) \in \delta, \quad \boldsymbol{y}_{\mathsf{acc}}[q_N] = 1.$$

Transition matrix and blocks We use $\boldsymbol{e}_q \in \{0,1\}^Q$ to represent the current state of a DFA. For a DFA $M = (Q, \boldsymbol{y}_{\mathsf{acc}}, \delta)$, its transition matrix is

$$\mathbf{M}(x)[q, q'] = \begin{cases} 1, & \text{if } ((q, x), q') \in \delta \\ 0, & \text{otherwise.} \end{cases}$$

For all $q \in [Q]$ and $x \in \{0,1\}$, consider $\boldsymbol{c}^\top \boldsymbol{e}_q^\top \mathbf{M}(x)$ — we have $\boldsymbol{c} \in \{0,1\}^Q$ and $\boldsymbol{c}[q'] = 1$ if and only if q' is a valid state after the DFA reads x in state q. Inductively, $\boldsymbol{e}_q^\top \mathbf{M}(x_1) \cdots \mathbf{M}(x_n)$ is a vector that counts the number of computation paths reaching each state starting from state q after reading $x_1, \ldots, x_n$. Let the *transition blocks* be $\mathbf{M}_x = \mathbf{M}(x)$ for $x \in \{0,1\}$, the $\mathbf{M}(x) = (1-x)\mathbf{M}_0 + x\mathbf{M}_1$. We arithmetize the computation of DFA by defining

$$M|_N(x) = \boldsymbol{e}_1^\top \prod_{i=1}^{N} ((1 - \boldsymbol{x}[i])\mathbf{M}_0 + \boldsymbol{x}[i]\mathbf{M}_1) \cdot \boldsymbol{y}_{\mathsf{acc}} \text{ over } \mathbb{Z}_p \text{ for } \boldsymbol{x} \in \mathbb{Z}_p^N.$$

AKGS **for DFA** Similar to the AKGS construction used in our FE scheme for Turing machines, the recursive mechanism for garbling the matrix multiplication yields a piecewise secure

AKGS for DFA. Let us consider the function class $\mathcal{F} = \{(M, 1^N, p) | M \text{ is an DFA}, p \text{ is prime}\}$, i.e., $M|_N$ is a function over $\mathbb{Z}_p$ and is represented as $(M, 1^N, p)$. The AKGS = (Garble, Eval) for $\mathcal{F}$ works as follows:

Garble$((M, 1^N, p), z, \beta)$ It takes input the DFA $(M, 1^N, p)$ and two secret integers $z, \beta \in \mathbb{Z}_p$. It computes the transition blocks $\mathbf{M}_0$ and $\mathbf{M}_1$ for M, sample $\boldsymbol{r}_0, \ldots, \boldsymbol{r}_N \leftarrow \mathbb{Z}_p^Q$, and defines the label functions:

$$\begin{aligned} L_{\mathsf{init}}(\boldsymbol{x}) &= \beta + \boldsymbol{e}_1^\top \boldsymbol{r}_0, \\ \text{for } i \in [N]: \quad (L_{i,q})_{q\in[Q]}(\boldsymbol{x}) &= -\boldsymbol{r}_{i-1} + ((1 - \boldsymbol{x}[i])\mathbf{M}_0 + \boldsymbol{x}[i]\mathbf{M}_1)\boldsymbol{r}_i \\ (L_{N+1,q})_{q\in[Q]} &= -\boldsymbol{r}_N + z\boldsymbol{y}_{\mathsf{acc}}. \end{aligned}$$

It collects the coefficients of these label functions and returns them as $(\boldsymbol{\ell}_{\mathsf{init}}, (\boldsymbol{\ell}_{i,q})_{i\in[N+1],q\in[Q]})$.

Eval$((M, 1^N, p), \boldsymbol{x}, \ell_{\mathsf{init}}, (\ell_{i,q})_{i\in[N+1],q\in[Q]})$ The evaluation procedure takes input string $\boldsymbol{x} \in \mathbb{Z}_p^N$ and the labels as input. It computes the transition blocks $\mathbf{M}_0, \mathbf{M}_1$ of M, sets $\boldsymbol{\ell}_i = (\ell_{i,q})_{q\in[Q]}$ for $i \in [N+1]$, and outputs the value

$$\ell_{\mathsf{init}} + \boldsymbol{e}_1^\top \sum_{i=1}^{N+1} \prod_{j=1}^{i-1} ((1 - \boldsymbol{x}[i])\mathbf{M}_0 + \boldsymbol{x}[i]\mathbf{M}_1) \cdot \boldsymbol{\ell}_i$$

We can similarly verify the correctness of the evaluation process and easily verify that the AKGS construction described above satisfies the linearity property and piecewise security as required.

Theorem 5 [62] *The* AKGS *construction for DFA described above is special piecewise secure with* L_{init} *being the first label function, the other label functions sorted in increasing order* i, *and the randomness sorted in the same order as the label functions.*

7.1 The construction

In this section, we only present the construction of our public key FE scheme for DFA that supports polynomial number of secret keys. We omit the description of the 1-key 1-ciphertext secure version of the scheme since it is a simpler variant of the public key counter part.

We now describe the construction of public key FE for UAWS for DFA $\mathsf{PK\text{-}UAWS}^{\mathsf{DFA}}_{(\mathsf{poly},1,1)}$ = (Setup, KeyGen, Enc, Dec). The Setup works similar to that of FE for UAWS for L (see Sect. 6.1). For the security analysis, we require some extra hidden subspaces the number of which can be determined while proving the security of the scheme similar to our public key FE for UAWS for L (see Sect. 6.2).

KeyGen(MSK, $(\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}})$): On input the master secret key MSK = $(\mathsf{IPFE.MSK}, \widetilde{\mathsf{IPFE.MSK}})$ and a function tuple $\boldsymbol{M} = (M_k)_{k\in\mathcal{I}_{\boldsymbol{M}}}$ indexed w.r.t. an index set $\mathcal{I}_{\boldsymbol{M}} \subset \mathbb{N}$ of arbitrary size , it parses $M_k = (Q_k, \boldsymbol{y}_k, \delta_k)$ $\forall k \in \mathcal{I}_{\boldsymbol{M}}$ and samples the set of elements

$$\left\{\alpha, \beta_k \leftarrow \mathbb{Z}_p \mid k \in \mathcal{I}_{\boldsymbol{M}}, \sum_k \beta_k = 0 \mod p\right\}.$$

It computes a secret key $\mathsf{IPFE.SK}_{\mathsf{pad}} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{\mathsf{pad}}]\!]_2)$ for the following vector $\boldsymbol{v}_{\mathsf{pad}}$:

vector	index$_1$	index$_2$	pad	init$^{\mathsf{pub}}$	rand$^{\mathsf{pub}}$	acc$^{\mathsf{pub}}$	tb$_0^{\mathsf{pub}}$	tb$_1^{\mathsf{pub}}$	the other indices
$\boldsymbol{v}_{\mathsf{pad}}$	0	0	α	0	0	0	0	0	0

For all $k \in \mathcal{I}_{\boldsymbol{M}}$, do the following:

1. For $M_k = (Q_k, \boldsymbol{y}_k, \delta_k)$, compute transition blocks $\mathbf{M}_{k,0}, \mathbf{M}_{k,1} \in \{0,1\}^{Q_k \times Q_k}$.
2. Sample independent random vector $\boldsymbol{r}_{k,f} \leftarrow \mathbb{Z}_p^{Q_k}$ and a random element $\pi_k \in \mathbb{Z}_p$.
3. For the following vector $\boldsymbol{v}_{k,\mathsf{init}}$, compute a secret key $\mathsf{IPFE.SK}_{k,\mathsf{init}} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{k,\mathsf{init}}]\!]_2)$:

vector	index_1	index_2	pad	$\mathsf{init}^{\mathsf{pub}}$	$\mathsf{rand}^{\mathsf{pub}}$	$\mathsf{acc}^{\mathsf{pub}}$	$\mathsf{tb}_0^{\mathsf{pub}}$	$\mathsf{tb}_1^{\mathsf{pub}}$	the other indices
$\boldsymbol{v}_{k,\mathsf{init}}$	π_k	$k \cdot \pi_k$	0	$\boldsymbol{r}_{k,f}[1]$	0	β_k	0	0	0

4. For each $q \in [Q_k]$, compute the following secret keys

$$\mathsf{IPFE.SK}_{k,q} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{k,q}]\!]_2) \quad \text{and}$$
$$\widetilde{\mathsf{IPFE.SK}}_{k,q} \leftarrow \mathsf{IPFE.KeyGen}(\widetilde{\mathsf{IPFE.MSK}}, [\![\widetilde{\boldsymbol{v}}_{k,q}]\!]_2)$$

where the vectors $\boldsymbol{v}_{k,q}, \widetilde{\boldsymbol{v}}_{k,q}$ are defined as follows:

vector	index_1	index_2	pad	$\mathsf{init}^{\mathsf{pub}}$	$\mathsf{rand}^{\mathsf{pub}}$	$\mathsf{acc}^{\mathsf{pub}}$	$\mathsf{tb}_0^{\mathsf{pub}}$	$\mathsf{tb}_1^{\mathsf{pub}}$	the other indices
$\boldsymbol{v}_{k,q}$	π_k	$k \cdot \pi_k$	0	0	$-\boldsymbol{r}_{k,f}[q]$	0	$(\mathbf{M}_{k,0}\boldsymbol{r}_{k,f})[q]$	$(\mathbf{M}_{k,1}\boldsymbol{r}_{k,f})[q]$	0

vector	index_1	index_2	$\mathsf{rand}^{\mathsf{pub}}$	$\mathsf{acc}^{\mathsf{pub}}$	the other indices
$\widetilde{\boldsymbol{v}}_{k,q}$	k	$k \cdot \pi_k$	$-\boldsymbol{r}_{k,f}[q]$	$\alpha \cdot \boldsymbol{y}_k[q]$	0

Finally, it returns the secret key as

$$\mathsf{SK}_{(\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}})} = \left((\boldsymbol{M}, \mathcal{I}_{\boldsymbol{M}}), \mathsf{IPFE.SK}_{\mathsf{pad}}, \left\{ \mathsf{IPFE.SK}_{k,\mathsf{init}}, \{\mathsf{IPFE.SK}_{k,q}, \widetilde{\mathsf{IPFE.SK}}_{k,q}\}_{q \in [Q_k]} \right\}_{k \in \mathcal{I}_{\boldsymbol{M}}} \right).$$

$\mathsf{Enc}(\mathsf{MPK}, \boldsymbol{x}, \boldsymbol{z})$: On input the master public key $\mathsf{MPK} = (\mathsf{IPFE.MPK}, \widetilde{\mathsf{IPFE.MPK}})$, a public attribute $\boldsymbol{x} \in \{0,1\}^N$ for some arbitrary $N \geq 1$, and the private attribute $\boldsymbol{z} \in \mathbb{Z}_p^n$ for some arbitrary $n \geq 1$, it samples $s \leftarrow \mathbb{Z}_p$ and compute a ciphertext $\mathsf{IPFE.CT}_{\mathsf{pad}} \leftarrow \mathsf{IPFE.Enc}(\mathsf{IPFE.MPK}, [\![\boldsymbol{u}_{\mathsf{pad}}]\!]_1)$ for the vector $\boldsymbol{u}_{\mathsf{pad}}$:

vector	index_1	index_2	pad	$\mathsf{init}^{\mathsf{pub}}$	$\mathsf{rand}^{\mathsf{pub}}$	$\mathsf{acc}^{\mathsf{pub}}$	$\mathsf{tb}_0^{\mathsf{pub}}$	$\mathsf{tb}_1^{\mathsf{pub}}$	the other indices
$\boldsymbol{u}_{\mathsf{pad}}$	0	0	s	0	0	0	0	0	0

Next, it does the following:

1. Sample a random vector $\boldsymbol{r}_{\boldsymbol{x}} \leftarrow \mathbb{Z}_p^{[0,N]}$.
2. For each $k \in [n]$, do the following:

   (a) Sample a random element $\rho_k \leftarrow \mathbb{Z}_p$.

   (b) Compute a ciphertext $\mathsf{IPFE.CT}_{k,\mathsf{init}} \leftarrow \mathsf{IPFE.SlotEnc}(\mathsf{IPFE.MPK}, [\![\boldsymbol{u}_{k,\mathsf{init}}]\!]_1)$ for the vector $\boldsymbol{u}_{k,\mathsf{init}}$:

vector	index_1	index_2	pad	$\mathsf{init}^{\mathsf{pub}}$	$\mathsf{rand}^{\mathsf{pub}}$	$\mathsf{acc}^{\mathsf{pub}}$	$\mathsf{tb}_0^{\mathsf{pub}}$	$\mathsf{tb}_1^{\mathsf{pub}}$	the other indices
$\boldsymbol{u}_{k,\mathsf{init}}$	$-k \cdot \rho_k$	ρ_k	0	$s \cdot \boldsymbol{r}_{\boldsymbol{x}}[0]$	0	s	0	0	$\perp$

   (c) For all $i \in [N]$, do the following:

      (i) Compute $\mathsf{IPFE.CT}_{k,i} \leftarrow \mathsf{IPFE.SlotEnc}(\mathsf{IPFE.MPK}, [\![\boldsymbol{u}_{k,i}]\!]_1)$ for the vector $\boldsymbol{u}_{k,i}$:

vector	index_1	index_2	pad	$\mathsf{init}^{\mathsf{pub}}$	$\mathsf{rand}^{\mathsf{pub}}$	$\mathsf{acc}^{\mathsf{pub}}$	$\mathsf{tb}_0^{\mathsf{pub}}$	$\mathsf{tb}_1^{\mathsf{pub}}$	the other indices
$\boldsymbol{u}_{k,i}$	$-k \cdot \rho_k$	ρ_k	0	0	$s \cdot \boldsymbol{r}_{\boldsymbol{x}}[i-1]$	0	$s \cdot \boldsymbol{r}_{\boldsymbol{x}}[i](1 - \boldsymbol{x}[i])$	$s \cdot \boldsymbol{r}_{\boldsymbol{x}}[i]\boldsymbol{x}[i]$	$\perp$

(d) For $i = N + 1$, compute $\widetilde{\mathsf{IPFE.CT}}_{k,N+1} \leftarrow \mathsf{IPFE.SlotEnc}(\widetilde{\mathsf{IPFE.MPK}}, [\![\widetilde{\boldsymbol{u}}_{k,N+1}]\!]_1)$ for the vector $\widetilde{\boldsymbol{u}}_{k,N+1}$:

vector	index_1	index_2	$\mathsf{rand}^{\mathsf{pub}}$	$\mathsf{acc}^{\mathsf{pub}}$	the other indices
$\widetilde{\boldsymbol{u}}_{k,N+1}$	$-k \cdot \rho_k$	ρ_k	$s \cdot \boldsymbol{r}_{\boldsymbol{x}}[N]$	$s \cdot \boldsymbol{z}[k]$	$\perp$

3. Finally, it returns the ciphertext as

$$\mathsf{CT}_{\boldsymbol{x}} = \left(\boldsymbol{x}, n, \mathsf{IPFE.CT}_{\mathsf{pad}}, \left\{\mathsf{IPFE.CT}_{k,\mathsf{init}}, \{\mathsf{IPFE.CT}_{k,i}\}_{i\in[N]}, \widetilde{\mathsf{IPFE.CT}}_{k,N+1}\right\}_{k\in[n]}\right).$$

Dec$(\mathsf{SK}_{(\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}})}, \mathsf{CT}_{\boldsymbol{x}})$: On input a secret key $\mathsf{SK}_{(\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}})}$ and a ciphertext $\mathsf{CT}_{\boldsymbol{x}}$, do the following:

1. Parse $\mathsf{SK}_{(\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}})}$ and $\mathsf{CT}_{(\boldsymbol{x},T,S)}$ as follows:

$$\mathsf{SK}_{(\boldsymbol{M},\mathcal{I}_{\boldsymbol{M}})} = \Bigg(\left((M_k)_{k\in\mathcal{I}_{\boldsymbol{M}}}, \mathcal{I}_{\boldsymbol{M}}\right), \mathsf{IPFE.SK}_{\mathsf{pad}}, \Big\{\mathsf{IPFE.SK}_{k,\mathsf{init}}, \{\mathsf{IPFE.SK}_{k,q}, \widetilde{\mathsf{IPFE.SK}}_{k,q}\}_{q\in[Q_k]}\Big\}_{k\in\mathcal{I}_{\boldsymbol{M}}}\Bigg), M_k = (Q_k, \boldsymbol{y}_k, \delta_k),$$

$$\mathsf{CT}_{\boldsymbol{x}} = \left(\boldsymbol{x}, n, \mathsf{IPFE.CT}_{\mathsf{pad}}, \left\{\mathsf{IPFE.CT}_{k,\mathsf{init}}, \{\mathsf{IPFE.CT}_{k,i}\}_{i\in[N]}, \widetilde{\mathsf{IPFE.CT}}_{k,N+1}\right\}_{k\in[n]}\right).$$

2. Output $\perp$, if $\mathcal{I}_{\boldsymbol{M}} \not\subset [n]$. Else, it proceeds to the next step.
3. Use the IPFE decryption to obtain $[\![\mu_{\mathsf{pad}}]\!]_{\mathrm{T}} \leftarrow \mathsf{IPFE.Dec}(\mathsf{IPFE.SK}_{\mathsf{pad}}, \mathsf{IPFE.CT}_{\mathsf{pad}})$.
4. For $k \in \mathcal{I}_{\boldsymbol{M}}, i \in [N]$, invoke the IPFE decryption to compute all label values as:

$$\begin{aligned}
&\forall k \in \mathcal{I}_{\boldsymbol{M}} : && [\![\ell_{k,\mathsf{init}}]\!]_{\mathrm{T}} = \mathsf{IPFE.Dec}(\mathsf{IPFE.SK}_{k,\mathsf{init}}, \mathsf{IPFE.CT}_{k,\mathsf{init}})\\
&\forall k \in \mathcal{I}_{\boldsymbol{M}}, i \in [N] : && [\![\ell_{k,i}]\!]_{\mathrm{T}} = \mathsf{IPFE.Dec}(\mathsf{IPFE.SK}_{k,q}, \mathsf{IPFE.CT}_{k,i})\\
&\forall k \in \mathcal{I}_{\boldsymbol{M}}, i = N+1 : && [\![\ell_{k,N+1}]\!]_{\mathrm{T}} = \mathsf{IPFE.Dec}(\widetilde{\mathsf{IPFE.SK}}_{k,q}, \widetilde{\mathsf{IPFE.CT}}_{k,N+1})
\end{aligned}$$

5. Next, invoke the AKGS evaluation procedure and obtain the combined value

$$[\![\mu]\!]_{\mathrm{T}} = \prod_{k\in\mathcal{I}_{\boldsymbol{M}}} \mathsf{Eval}\left(\left(M_k, 1^N, p\right), \boldsymbol{x}, [\![\ell_{k,\mathsf{init}}]\!]_{\mathrm{T}}, \left\{[\![\ell_{k,i}]\!]_{\mathrm{T}}\right\}_{i\in[N+1]}\right)$$

6. Finally, it returns μ' such that $[\![\mu]\!]_{\mathrm{T}} = ([\![\mu_{\mathsf{pad}}]\!]_{\mathrm{T}})^{\mu'}$, where $g_{\mathrm{T}} = e(g_1, g_2)$. Similar to [8], we assume that the desired attribute-weighted sum lies within a specified polynomial-sized domain so that μ' can be searched via brute-force.

The correctness of our $\mathsf{PK\text{-}UAWS}^{\mathsf{DFA}}_{(\mathsf{poly},1,1)}$ can be shown similarly to our public key FE scheme for L. We state the following corollary about the adaptive simulation security of the scheme. The corollary can be proved similar to the proof of the Theorem 4.

Corollary 1 *Assuming the* SXDH *assumption holds in* $\mathcal{G}$ *and the* IPFE *is function hiding secure, the above construction of* 1-Slot FE *for* UAWS *for DFA is adaptively simulation secure.*

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availability Data Deposition Information: Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Conflict of interest The authors have no relevant financial or non-financial conflict of interests to disclose.

References

1. Abdalla M., Benhamouda F., Gay R.: From single-input to multi-client inner-product functional encryption. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 552–582. Springer (2019).
2. Abdalla M., Benhamouda F., Kohlweiss M., Waldner H.: Decentralizing inner-product functional encryption. In: IACR International Workshop on Public Key Cryptography, pp. 128–157. Springer (2019).
3. Abdalla M., Bourse F., De Caro A., Pointcheval D.: Simple functional encryption schemes for inner products. In: PKC 2015, pp. 733–751. Springer (2015).
4. Abdalla M., Bourse F., Marival H., Pointcheval D., Soleimanian A., Waldner H.: Multi-client inner-product functional encryption in the random-oracle model. In: International Conference on Security and Cryptography for Networks, pp. 525–545. Springer (2020).
5. Abdalla M., Catalano D., Fiore D., Gay R., Ursu B.: Multi-input functional encryption for inner products: function-hiding realizations and constructions without pairings. In: CRYPTO 2018, pp. 597–627. Springer (2018).
6. Abdalla M., Catalano D., Gay R., Ursu B.: Inner-product functional encryption with fine-grained access control. IACR Cryptology ePrint Archive, Report 2020/577 (2020).
7. Abdalla M., Gay R., Raykova M., Wee H.: Multi-input inner-product functional encryption from pairings. In: CRYPTO 2017, pp. 601–626. Springer (2017).
8. Abdalla M., Gong J., Wee H.: Functional encryption for attribute-weighted sums from k-Lin. In: CRYPTO 2020, pp. 685–716. Springer (2020).
9. Agrawal S., Goyal R., Tomida J.: Multi-input quadratic functional encryption from pairings. In: Annual International Cryptology Conference, pp. 208–238. Springer (2021).
10. Agrawal S., Libert B., Maitra M., Titiu R.: Adaptive simulation security for inner product functional encryption. In: PKC 2020, pp. 34–64. Springer (2020).
11. Agrawal S., Libert B., Stehlé D.: Fully secure functional encryption for inner products, from standard assumptions. In: CRYPTO 2016, pp. 333–362. Springer (2016).
12. Agrawal S., Maitra M.: Fe and io for turing machines from minimal assumptions. In: Theory of Cryptography: 16th International Conference, TCC 2018, Panaji, India, November 11–14, 2018, Proceedings, Part II, pp. 473–512. Springer-Verlag (2018).
13. Agrawal S., Maitra M., Vempati N.S., Yamada S.: Functional encryption for turing machines with dynamic bounded collusion from LWE. In: CRYPTO 2021, pp. 239–269. Springer (2021).
14. Agrawal S., Rosen A.: Functional encryption for bounded collusions, revisited. In: Theory of Cryptography Conference, pp. 173–205. Springer (2017).
15. Ananth P., Brakerski Z., Segev G., Vaikuntanathan V.: From selective to adaptive security in functional encryption. In: Annual Cryptology Conference, pp. 657–677. Springer (2015).
16. Ananth P., Sahai A.: Functional encryption for turing machines. In: TCC 2016, pp. 125–153. Springer (2016).
17. Ananth P., Vaikuntanathan V.: Optimal bounded-collusion secure functional encryption. In: Theory of Cryptography Conference, pp. 174–198. Springer (2019).
18. Applebaum B., Ishai Y., Kushilevitz E.: How to garble arithmetic circuits. In: FOCS 2011, pp. 120–129. IEEE Computer Society (2011).
19. Attrapadung N.: Dual system encryption framework in prime-order groups via computational pair encodings. In: ASIACRYPT 2016, pp. 591–623. Springer (2016).
20. Badrinarayanan S., Goyal V., Jain A., Sahai A.: Verifiable functional encryption. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 557–587. Springer (2016).
21. Baltico C.E.Z., Catalano D., Fiore D., Gay R.: Practical functional encryption for quadratic functions with applications to predicate encryption. In: CRYPTO 2017, pp. 67–98. Springer (2017).

22. Boneh D., Gentry C., Gorbunov S., Halevi S., Nikolaenko V., Segev G., Vaikuntanathan V., Vinayagamurthy D.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: EUROCRYPT 2014, pp. 533–556. Springer (2014).
23. Boneh D., Lewi K., Raykova M., Sahai A., Zhandry M., Zimmerman J.: Semantically secure order-revealing encryption: multi-input functional encryption without obfuscation. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 563–594. Springer (2015).
24. Boneh D., Sahai A., Waters B.: Functional encryption: definitions and challenges. In: TCC 2011, pp. 253–273. Springer (2011).
25. Brakerski Z., Chandran N., Goyal V., Jain A., Sahai A., Segev G.: Hierarchical functional encryption. In: 8th Innovations in Theoretical Computer Science Conference (ITCS 2017). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2017).
26. Brakerski Z., Komargodski I., Segev G.: Multi-input functional encryption in the private-key setting: stronger security from weaker assumptions. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 852–880. Springer (2016).
27. Brakerski Z., Segev G.: Function-private functional encryption in the private-key setting. J. Cryptol. **31**(1), 202–225 (2018).
28. Caro A.D., Iovino V., O'Neill A.: Deniable functional encryption. In: Public-Key Cryptography—PKC 2016, pp. 196–222. Springer (2016).
29. Castagnos G., Laguillaumie F., Tucker I.: Practical fully secure unrestricted inner product functional encryption modulo p. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 733–764. Springer (2018).
30. Chen J., Gay R., Wee H.: Improved dual system ABE in prime-order groups via predicate encodings. In: EUROCRYPT 2015, pp. 595–624. Springer (2015).
31. Chen J., Gong J., Kowalczyk L., Wee H.: Unbounded ABE via bilinear entropy expansion, revisited. In: EUROCRYPT 2018, pp. 503–534. Springer (2018).
32. Chotard J., Dufour Sans E., Gay R., Phan D.H., Pointcheval D.: Decentralized multi-client functional encryption for inner product. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 703–732. Springer (2018).
33. Chotard J., Dufour-Sans E., Gay R., Phan D.H., Pointcheval D.: Dynamic decentralized functional encryption. In: Annual International Cryptology Conference, pp. 747–775. Springer (2020).
34. Chung K.M., Katz J., Zhou H.S.: Functional encryption from (small) hardware tokens. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 120–139. Springer (2013).
35. Ciampi M., Siniscalchi L., Waldner H.: Multi-client functional encryption for separable functions. In: IACR International Conference on Public-Key Cryptography, pp. 724–753. Springer (2021).
36. Datta P., Dutta R., Mukhopadhyay S.: Functional encryption for inner product with full function privacy. In: PKC 2016, pp. 164–195. Springer (2016).
37. Datta P., Okamoto T., Takashima K.: Adaptively simulation-secure attribute-hiding predicate encryption. In: ASIACRYPT 2018, pp. 640–672. Springer (2018).
38. Datta P., Pal T.: (Compact) adaptively secure FE for attribute-weighted sums from k-Lin. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 434–467. Springer (2021).
39. Datta P., Pal T., Takashima K.: Compact FE for unbounded attribute-weighted sums for Logspace from SXDH. In: Advances in Cryptology—ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part I, pp. 126–159. Springer (2023).
40. Dufour-Sans E., Pointcheval D.: Unbounded inner-product functional encryption with succinct keys. In: ACNS 2019, pp. 426–441. Springer (2019).
41. Garg R., Goyal R., Lu G., Waters B.: Dynamic collusion bounded functional encryption from identity-based encryption. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 736–763. Springer (2022).
42. Garg S., Gentry C., Halevi S., Raykova M., Sahai A., Waters B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. SIAM J. Comput. **45**(3), 882–929 (2016).
43. Garg S., Gentry C., Halevi S., Zhandry M.: Functional encryption without obfuscation. In: Theory of Cryptography Conference, pp. 480–511. Springer (2016).
44. Garg S., Srinivasan A.: Single-key to multi-key functional encryption with polynomial loss. In: Hirt M., Smith A.D. (eds.) Theory of Cryptography—14th International Conference, TCC 2016-B, Beijing, China, October 31–November 3, 2016, Proceedings, Part II, Lecture Notes in Computer Science, vol. 9986, pp. 419–442 (2016). https://doi.org/10.1007/978-3-662-53644-5_16.

45. Goldwasser S., Gordon S.D., Goyal V., Jain A., Katz J., Liu F.H., Sahai A., Shi E., Zhou H.S.: Multi-input functional encryption. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 578–602. Springer (2014).
46. Goldwasser S., Kalai Y., Popa R.A., Vaikuntanathan V., Zeldovich N.: Reusable garbled circuits and succinct functional encryption. In: STOC 2013, pp. 555–564. ACM (2013).
47. Goldwasser S., Kalai Y.T., Popa R.A., Vaikuntanathan V., Zeldovich N.: How to run turing machines on encrypted data. In: Annual Cryptology Conference, pp. 536–553. Springer (2013).
48. Gorbunov S., Vaikuntanathan V., Wee H.: Functional encryption with bounded collusions via multi-party computation. In: CRYPTO 2012, pp. 162–179. Springer (2012).
49. Goyal V., Jain A., Koppula V., Sahai A.: Functional encryption for randomized functionalities. In: Theory of Cryptography Conference, pp. 325–351. Springer (2015).
50. Iovino V., Żebroski K.: Simulation-based secure functional encryption in the random oracle model. In: International Conference on Cryptology and Information Security in Latin America, pp. 21–39. Springer (2015).
51. Ishai Y., Wee H.: Partial garbling schemes and their applications. In: ICALP 2014, pp. 650–662. Springer (2014).
52. Jain A., Lin H., Sahai A.: Indistinguishability obfuscation from well-founded assumptions. In: Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, pp. 60–73 (2021).
53. Jain A., Lin H., Sahai A.: Indistinguishability obfuscation from LPN over, DLIN, and PRGS in NC. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 670–699. Springer (2022).
54. Kim S., Lewi K., Mandal A., Montgomery H., Roy A., Wu D.J.: Function-hiding inner product encryption is practical. In: SCN 2018, pp. 544–562. Springer (2018).
55. Komargodski I., Segev G., Yogev E.: Functional encryption for randomized functionalities in the private-key setting from minimal assumptions. J. Cryptol. **31**(1), 60–100 (2018).
56. Lai Q., Liu F.H., Wang Z.: New lattice two-stage sampling technique and its applications to functional encryption—stronger security and smaller ciphertexts. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 498–527. Springer (2021).
57. Lewko A., Okamoto T., Sahai A., Takashima K., Waters B.: Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: EUROCRYPT 2010, pp. 62–91. Springer (2010).
58. Lewko A.B., Waters B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: TCC 2010, pp. 455–479. Springer (2010).
59. Lewko A.B., Waters B.: Unbounded HIBE and attribute-based encryption. In: EUROCRYPT 2011, pp. 547–567. Springer (2011).
60. Libert B., Ţiţiu R.: Multi-client functional encryption for linear functions in the standard model from LWE. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 520–551. Springer (2019).
61. Lin H.: Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGS. In: CRYPTO 2017, pp. 599–629. Springer (2017).
62. Lin H., Luo J.: Compact adaptively secure ABE from k-Lin: beyond NC^1 and towards NL. In: EUROCRYPT 2020, pp. 247–277. Springer (2020).
63. Lin H., Tessaro S.: Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. In: Annual International Cryptology Conference, pp. 630–660. Springer (2017).
64. Lin H., Vaikuntanathan V.: Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In: FOCS 2016, pp. 11–20. IEEE (2016).
65. Liu X., Liu S., Han S., Gu D.: Tightly CCA-secure inner product functional encryption scheme. Theor. Comput. Sci. **898**, 1–19 (2022).
66. Mera J.M.B., Karmakar A., Marc T., Soleimanian A.: Efficient lattice-based inner-product functional encryption. In: IACR International Conference on Public-Key Cryptography, pp. 163–193. Springer (2022).
67. Okamoto T., Takashima K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: CRYPTO 2010, pp. 191–208. Springer (2010).
68. Okamoto T., Takashima K.: Fully secure unbounded inner-product and attribute-based encryption. In: ASIACRYPT 2012, pp. 349–366. Springer (2012).
69. O'Neill A.: Definitional issues in functional encryption. IACR Cryptology ePrint Archive, Report 2010/556 (2010).
70. Tomida J., Abe M., Okamoto T.: Efficient functional encryption for inner-product values with full-hiding security. In: ICS 2016, pp. 408–425. Springer (2016).

71. Tomida J., Takashima K.: Unbounded inner product functional encryption from bilinear maps. Jpn. J. Ind. Appl. Math. **37**(3), 723–779 (2020).
72. Wang Z., Fan X., Liu F.H.: FE for inner products and its application to decentralized ABE. In: PKC 2019, pp. 97–127. Springer (2019).
73. Waters B.: Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In: CRYPTO 2009, pp. 619–636. Springer (2009).
74. Waters B.: A punctured programming approach to adaptively secure functional encryption. In: Annual Cryptology Conference, pp. 678–697. Springer (2015).
75. Wee H.: Dual system encryption via predicate encodings. In: TCC 2014, pp. 616–637. Springer (2014).
76. Wee H.: Attribute-hiding predicate encryption in bilinear groups, revisited. In: TCC 2017, pp. 206–233. Springer (2017).
77. Wee H.: Functional encryption for quadratic functions from k-Lin, revisited. In: TCC 2020, pp. 210–228. Springer (2020).