Non-Interactive Zero-Knowledge Functional Proofs

Gongxian Zeng[1], Junzuo Lai[2], Zhengan Huang[1], Linru Zhang[3], Xiangning Wang[3], Kwok-Yan Lam[3], Huaxiong Wang[3], and Jian Weng[2]

[1] Peng Cheng Laboratory, Shenzhen, China
gxzeng@cs.hku.hk, zhahuang.sjtu@gmail.com
[2] College of Information Science and Technology, Jinan University, Guangzhou, China
{laijunzuo,cryptjweng}@gmail.com
[3] Nanyang Technological University, Singapore
{linru.zhang,xiangning.wang,kwokyan.lam,hxwang}@ntu.edu.sg

Abstract. In this paper, we consider to generalize NIZK by empowering a prover to share a witness in a fine-grained manner with verifiers. Roughly, the prover is able to authorize a verifier to obtain extra information of witness, i.e., besides verifying the truth of the statement, the verifier can additionally obtain certain function of the witness from the accepting proof using a secret functional key provided by the prover.
To fulfill these requirements, we introduce a new primitive called *non-interactive zero-knowledge functional proofs (fNIZKs)*, and formalize its security notions. We provide a generic construction of fNIZK for any NP relation $\mathcal{R}$, which enables the prover to share any function of the witness with a verifier. For a widely-used relation about set membership proof (implying range proof), we construct a concrete and efficient fNIZK, through new building blocks (set membership encryption and dual inner-product encryption), which might be of independent interest.

Keywords: non-interactive zero knowledge proof, set membership proof, range proof, inner-product encryption

1 Introduction

The zero-knowledge (ZK) proof system [20] is an interactive protocol in which a prover convinces a verifier of the truth of a statement without disclosing any additional information. A non-interactive zero-knowledge (NIZK) proof [1] is a type of ZK proof without any interactions with a verifier. NIZKs have found numerous applications in cryptography, including but not limited to secure public key encryption resilient against chosen-ciphertext attacks [28], group/ring signatures [9,7], anonymous credentials [7], multi-party computations [19], and some applications in blockchain such as privacy preserving coins (e.g., Zcash [29]), zero knowledge virtual machine (e.g., zkEVM [34]) and blockchain-based e-voting [22].

In this paper, our objective is to generalize the concept of NIZK by enabling the prover to share a witness in a fine-grained manner with verifiers. Specifically, the prover is granted the ability to authorize a verifier to access additional information of the witness. This means that, in addition to verifying the truth of the statement, the verifier can also gain insights into certain functions of the witness using a secret key provided by the prover. To address these requirements, we propose a new type of NIZKs called *non-interactive zero-knowledge functional proofs (fNIZKs)*.

Our contributions. We initiate the study of fNIZK. The specific contributions are outlined as follows:

1. We present a formal definition and security notions of non-interactive zero-knowledge functional proof (fNIZK).
2. We provide a generic construction of fNIZK for any NP relation $\mathcal{R}$, which enables the prover to share any function of the witness with a verifier.
3. For a widely-used relation about set membership proof (implying range proof), we construct a concrete and efficient fNIZK, called set membership functional proof (fSMP).

Primitive of fNIZK. A fNIZK scheme consists of seven algorithms: Setup, Prove, Verify, UKGen, FKGen, $\mathsf{CheckKey}$ and $\mathsf{Extract}$. Roughly, $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ are similar to those of NIZK, except that Prove and Verify also input the prover's public key, which is generated by UKGen. The prover invokes FKGen to generate a secret key for some function (secret functional key) and distribute it to a verifier. With this key, the verifier can call $\mathsf{Extract}$ to extract the function of the witness from an accepting proof. The validity of the secret functional key can be checked via $\mathsf{CheckKey}$.

If there is no restrictions on the extracting capability of keys, then a verifier with a secret functional key can learn a function of the witnesses in *all* accepting proofs generated by the prover. To address the above issue, we introduce *labels* in some of the above algorithms.

Concretely, in fNIZK, the FKGen algorithm, whose input includes a label $\tau_{\mathtt{f}}$, the secret key sk of the prover, and a function f, generates a secret functional key $sk_{f,\tau_{\mathtt{f}}}$. Similarly, the Prove algorithm, whose input includes a label $\tau_{\mathtt{p}}$, the public key pk, and a statement-witness pair (x, w), to generate a proof π. The extraction algorithm $\mathsf{Extract}$ can output $f(w)$ from π, only if $\mathsf{P}(\tau_{\mathtt{p}}, \tau_{\mathtt{f}}) = 1$ for some predicate P.

The security properties of fNIZK contains completeness, functional knowledge, adaptive soundness and zero knowledge. Soundness is similar to that of NIZK. Other properties are listed below.

1. Completeness of fNIZK has three requirements. Firstly, any proof π generated by Prove should be verified successfully by Verify. Secondly, any normally generated secret functional key $sk_{f,\tau_{\mathtt{f}}}$ for $(f, \tau_{\mathtt{f}})$ should pass the verification of $\mathsf{CheckKey}$. Thirdly, for normally generated proof π (associated with $\tau_{\mathtt{p}}$) and normally generated secret functional key $sk_{f,\tau_{\mathtt{f}}}$ (for $(f, \tau_{\mathtt{f}})$), if $\mathsf{P}(\tau_{\mathtt{p}}, \tau_{\mathtt{f}}) = 1$, $\mathsf{Extract}$ should extract $f(w)$ from π.

2. The functional knowledge property requires that, in general, if the verifier accepts a proof π associated with a label $\tau_{\mathtt{p}}$ (here π *does not have to be normally generated*), then he/she can be convinced that a function of some witness can be extracted from the proof (with the help of the secret functional key associated with label $\tau_{\mathtt{f}}$ satisfying $\mathsf{P}(\tau_{\mathtt{p}}, \tau_{\mathtt{f}}) = 1$).
3. The zero knowledge requires that except for the fact of the truth of the statement and the functions (authorized by the prover) of witness, the verifier cannot obtain any other information about the witness from an accepting proof.

Generic construction of fNIZK. Based on NIZKs and functional encryption (FE) [26,2], we provide a generic construction of fNIZK for any NP relation $\mathcal{R}$, which enables the prover to share any function (from a function family $\mathbb{F}$) of the witness with a verifier.

When generating a proof for a valid statement-witness pair (x, w), Prove firstly encrypts the witness using the underlying FE scheme, then utilizes the NIZK to prove that "$(x, w) \in \mathcal{R}$ and the well-formedness of the ciphertext", and finally outputs a proof including the ciphertext and the NIZK proof. Extract can be implemented by calling the decryption of FE. We require that the underlying FE supports function family $\widehat{\mathbb{F}}$, where a function $\widehat{f}$ belongs to $\widehat{\mathbb{F}}$, if and only if there exists $(f \in \mathbb{F}, \tau_{\mathtt{f}})$ satisfying

$$\widehat{f}(w, \tau_{\mathtt{p}}) = \begin{cases} f(w) & \text{if } \mathsf{P}(\tau_{\mathtt{p}}, \tau_{\mathtt{f}}) = 1 \\ \perp & \text{if } \mathsf{P}(\tau_{\mathtt{p}}, \tau_{\mathtt{f}}) = 0 \end{cases}$$

The secret functional key generated by FKGen also contains a NIZK proof, which enables the verifier to check the validity of the secret functional key, via CheckKey.

The security properties of this fNIZK construction are derived from the properties of the FE scheme and the NIZK schemes.

Concrete and efficient construction of fSMP. Set membership proof (SMP) [5] enables a prover to convince a verifier that a digitally committed value belongs to a specified public set. A special case of SMP is range proof [5,4,13], where the public set is an integer range. SMPs are widely utilized as building blocks in various cryptographic schemes such as anonymous credentials [7,31], Zcash [29], and e-cash [6].

Due to the extensive application value of SMPs, we provide a concrete and efficient fNIZK for relation about set membership proof, called set membership functional proof (fSMP).

In our fSMP, Prove outputs a proof associated with label $\tau_{\mathtt{p}}$ to demonstrate that the committed value $w \in \Phi$ where Φ is a public set, and a verifier with a secret functional key for $(\Phi_S, \tau_{\mathtt{f}})$ can additionally check whether $w \in \Phi_S$ or not (where $\Phi_S \subset \Phi$) from the proof, when $\mathsf{P}(\tau_{\mathtt{p}}, \tau_{\mathtt{f}}) = 1$.

To construct a fSMP, we propose a new primitive, called *set membership encryption* (SME), which is a variant of public-key encryption, including Setup, KGen, Enc and Query. Roughly, Enc takes a set Φ, a message $w \in \Phi$ and a label

$\tau_{\mathtt{p}}$ as input to generate a ciphertext c. The query algorithm Query takes as input c and a secret functional key for $(\Phi_S \subset \Phi, \tau_{\mathtt{f}})$ generated by KGen, and outputs a bit. We require that when $\mathsf{P}(\tau_{\mathtt{p}}, \tau_{\mathtt{f}}) = 1$, Query outputs 1 if and only if $w \in \Phi_S$. We say that a SME supports Sigma protocols, if there exists a Sigma protocol to prove the well-formedness of a SME ciphertext. Then we show a generic framework of constructing fSMP from a SME and a commitment scheme that both support Sigma protocols[4].

With the help of a new building block called *dual inner-product encryption* (dual IPE), we present a generic construction of SME. Dual IPE is a special two-level hierarchical IPE (2-HIPE) [24,21,25] without delegation capability. In terms of attribute-hiding, our dual IPE requires the first-level vector to be fully attribute-hiding, without requiring the second-level vector to be hidden.

We provide an efficient instantiation of dual IPE, utilizing the techniques in IPE [10,33,12], based on the k-LIN assumption.

When plugging the dual IPE instantiation ($k = 1$) into the generic construction of SME from dual IPE, we can obtain a concrete and efficient SME supporting Sigma protocols. Further, incorporating Pedersen commitment [27], we achieve a concrete and efficient fSMP.

Finally, we improve the size of the proof of fSMP. Note that fSMP contains a NIZK proof, which derives from the Sigma protocols about the SME and Pedersen commitment. We utilize the self-stacking technique [18] to achieve logarithmic size of the NIZK proof, i.e., $O((\log l_1) \cdot \mathsf{poly}(\lambda)) = O(\log l_1)$, where $l_1 = |\Phi|$ and λ is the security parameter.

Applications of fNIZK. In the following, we present examples that illustrate the promising applications of fNIZK.

A scenario where fNIZK can work effectively is in supervision[5], such as anti-money laundering. Typically, individuals generate zero knowledge proofs for their regular activities, such as transferring privacy-preserving cryptocurrencies. In certain cases, the authority may authorize a specific department or institution for supervision purposes. For instance, the authority is able to issue a secret functional key that is only applicable to proofs generated during specific periods (facilitated by labels). Consequently, the department or institution can obtain the specified information from the proofs using the authorized key. This enables them to determine whether a user has violated the rules within certain periods, while ensuring that other information about the witnesses in the proofs remains undisclosed.

In blockchain-based auction systems, before participation in auctions, users lock a certain amount of coins by transferring them to the auction platform, accompanied by a NIZK proof that verifies the validity of the coins in a zero-knowledge manner. However, when users intend to participate in specific auc-

[4] A commitment scheme supports Sigma protocols, if there exists a Sigma protocol to prove the well-formedness of a commitment.

[5] In this scenario, the authority generates a public key and a secret key, and all users utilize the public key to generate or verify proofs. The secret functional keys are generated by the authority, using the secret key.

tions, the auction platform needs to verify if the amount of coins meets the minimum deposit requirement. Without the use of fNIZK, in order to maintain the privacy of the exact amount of coins, users would need to generate NIZK proofs for auctions with different minimum deposit requirements. By employing fNIZK, users only need to generate the proof once when transferring coins to the platform. Subsequently, they can generate secret keys associated with range functions (e.g., greater than the minimum deposit requirement) for the auction organizer, each time they wish to join particular auctions. The auction organizer can extract the function about the amount to verify if the user meets the requirements for participation, while keeping other information about the amount of coins private.

Another application example is anonymous attribute-based credential [7]. Users who obtain credentials for specific attribute sets are required to show different proofs to different verifiers (e.g., service providers) in order to demonstrate possession of a valid credential that satisfies the access policies set by the service providers. We point out that fNIZK offers an alternative approach for constructing anonymous attribute-based credential systems. Firstly, taking all the attributes and the credentials as witness, the user can utilize fNIZK to generate a proof to demonstrate the well-formedness of a valid credential for a specific attribute set without disclosing any attributes. Then, when the user needs to show proofs for different service providers, he/she can simply send different secret keys for different functions associated with the access policies to different service providers, instead of generating multiple NIZK proofs. This approach could reduce computational overhead.

Roadmap. The remaining sections of this paper are structured as follows: Sec. 2 provides a review of the preliminaries. In Sec. 3, we present the formal definitions of syntax and security notions pertaining to fNIZK. A generic construction of fNIZK is presented in Sec. 4. Furthermore, we delve into the specific case of the set membership proof and introduce the construction of set membership functional proof (fSMP) in Sec. 5.

2 Preliminaries

Notations. Throughout this paper, let λ denote the security parameter. For any $k \in \mathbb{N}$, let $[k] := \{1, 2, \cdots, k\}$. For a finite set S, we denote by $|S|$ the number of elements in S, and denote by $a \leftarrow S$ the process of uniformly sampling a from S. For a distribution X, we denote by $a \leftarrow X$ the process of sampling a from X. For any probabilistic polynomial-time (PPT) algorithm Alg, let $\mathcal{RS}_{\mathsf{Alg}}$ be the randomness space of Alg. We write $\mathsf{Alg}(x; r)$ for the process of Alg on input x with inner randomness $r \in \mathcal{RS}_{\mathsf{Alg}}$, and use $y \leftarrow \mathsf{Alg}(x)$ to denote the process of running Alg on input x with $r \leftarrow \mathcal{RS}_{\mathsf{Alg}}$, and assigning y the result. We write $\mathsf{negl}(\lambda)$ to denote a negligible function in λ and write $\mathsf{poly}(\lambda)$ to denote a polynomial.

For a polynomial-time relation $\mathcal{R} \subset \mathcal{X} \times \mathcal{W}$, where $\mathcal{X}$ is the statement space and $\mathcal{W}$ is the witness space, we say that w is a witness for x if $(x, w) \in \mathcal{R}$. We denote the language associated with $\mathcal{R}$ as $\mathcal{L}_{\mathcal{R}} = \{x \mid \exists w : (x, w) \in \mathcal{R}\}$.

Bold lower-case letters denote vectors, e.g., $\mathbf{a} = (a_1, \ldots, a_n)$ is a n-dimension vector, and usually the number of dimensions can be inferred from the context. Let $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i \in [n]} a_i \cdot b_i$ denotes the inner product between two vectors $\mathbf{a}$ and $\mathbf{b}$. Bold upper-case letters denote matrices, e.g., $\mathbf{B} \in \mathbb{Z}_p^{n_1 \times n_2}$ is an $n_1 \times n_2$ matrix. We use $\mathbf{I}_n$ to denote the $n \times n$ identity matrix. For simplicity, we sometimes write $\mathbf{I}$ to denote the identity matrix when n is given in the text.

Due to space limitations, we place the other preliminaries in the appendices. Specifically, the definitions of NIZK and Sigma protocols (including the stackable Sigma protocols) are recalled in Appendix A. The definitions of functional encryption and inner-product encryption are recalled in Appendix B and Appendix C, respectively. The definition of commitment is placed in Appendix D.

3 Non-interactive zero-knowledge functional proof

In this section, we introduce a primitive called *non-interactive zero-knowledge functional proof (fNIZK)*, and formalize its security notions. Generally speaking, fNIZK offers the functionalities of NIZK, while also enabling a verifier with a specific secret key, provided by the prover, to extract specific information about the witness from the accepting proof.

Definition 1. (fNIZK). *Let $\mathcal{L}_{\mathcal{R}}$ be an* NP *language associated with an* NP *relation $\mathcal{R}$. Let $\mathbb{F}$ be a function family, and $\mathcal{T}$ be the label space. Let $\mathsf{P} : \mathcal{T} \times (\mathcal{T} \cup \{*\}) \to \{0, 1\}$ be a predicate function satisfying $\mathsf{P}(\tau, *) = 1$ for all $\tau \in \mathcal{T}$. A* non-interactive zero-knowledge functional proof (fNIZK proof) *for $\mathcal{L}_{\mathcal{R}}$, $\mathbb{F}$, $\mathcal{T}$ and* P *consists of a tuple of seven efficient algorithms* $\mathsf{fNIZK} = (\mathsf{Setup}, \mathsf{UKGen}, \mathsf{FKGen}, \mathsf{CheckKey}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Extract})$.

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$*: On input the security parameter λ, the setup algorithm outputs a common reference string* crs.
- $\mathsf{UKGen}(\mathsf{crs}) \to (pk, sk)$*: On input a common reference string* crs*, the user key generation algorithm outputs a public key pk and a secret key sk.*
- $\mathsf{FKGen}(\mathsf{crs}, pk, sk, f, \tau_{\mathtt{f}}) \to sk_{f,\tau_{\mathtt{f}}}$*: On input a common reference string* crs*, a user key pair (pk, sk), a function $f \in \mathbb{F}$ and a label $\tau_{\mathtt{f}} \in \mathcal{T} \cup \{*\}$, the secret functional key generation algorithm outputs a secret functional key $sk_{f,\tau_{\mathtt{f}}}$. We assume that $sk_{f,\tau_{\mathtt{f}}}$ implicitly includes the information of f and $\tau_{\mathtt{f}}$.*
- $\mathsf{CheckKey}(\mathsf{crs}, pk, f, \tau_{\mathtt{f}}, sk_{f,\tau_{\mathtt{f}}}) \to b$*: On input a common reference string* crs*, a public key pk, a function $f \in \mathbb{F}$ and a label $\tau_{\mathtt{f}} \in \mathcal{T} \cup \{*\}$ and a secret functional key $sk_{f,\tau_{\mathtt{f}}}$, the checking algorithm outputs a bit $b \in \{0, 1\}$.*
- $\mathsf{Prove}(\mathsf{crs}, pk, \tau_{\mathtt{p}}, x, w) \to \pi$*: On input a common reference string* crs*, a public key pk, a label $\tau_{\mathtt{p}} \in \mathcal{T}$, a statement x and a witness w, the proving algorithm outputs a proof π. We assume that there exists an efficient algorithm Ext_τ such that $\tau_{\mathtt{p}} \leftarrow \mathsf{Ext}_\tau(\pi)$.*

- $\mathsf{Verify}(\mathsf{crs}, pk, x, \pi) \to b$: *On input the common reference string* crs*, a public key* pk*, a statement* x *and a proof* π*, the verification algorithm outputs a bit* $b \in \{0,1\}$*.*
- $\mathsf{Extract}(\mathsf{crs}, x, \pi, sk_{f,\tau_\mathtt{f}}) \to y$: *On input a common reference string* crs*, a statement* x*, a proof* π *and a secret functional key* $sk_{f,\tau_\mathtt{f}}$ *(for* f *and* $\tau_\mathtt{f}$*), the extraction algorithm outputs* y*.*

Moreover, fNIZK *should satisfy the following properties:*

1. ***Completeness.*** *For any* $(x,w) \in \mathcal{R}$*, any* $f \in \mathbb{F}$*, any* $\tau_\mathtt{p} \in \mathcal{T}$ *and any* $\tau_\mathtt{f} \in \mathcal{T} \cup \{*\}$*,*

$$\Pr\left[\begin{array}{l}\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda),\ (pk,sk) \leftarrow \mathsf{UKGen}(\mathsf{crs}) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, pk, \tau_\mathtt{p}, x, w)\end{array} : \mathsf{Verify}(\mathsf{crs}, pk, x, \pi) = 1\right] \geq 1 - \mathsf{negl}(\lambda),$$

$$\Pr\left[\begin{array}{l}\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda),\ (pk,sk) \leftarrow \mathsf{UKGen}(\mathsf{crs}) \\ sk_{f,\tau_\mathtt{f}} \leftarrow \mathsf{FKGen}(\mathsf{crs}, pk, sk, f, \tau_\mathtt{f})\end{array} : \mathsf{CheckKey}(\mathsf{crs}, pk, f, \tau_\mathtt{f}, sk_{f,\tau_\mathtt{f}}) = 1\right] \geq 1 - \mathsf{negl}(\lambda),$$

$$\Pr\left[\begin{array}{l}\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ (pk,sk) \leftarrow \mathsf{UKGen}(\mathsf{crs}) \\ sk_{f,\tau_\mathtt{f}} \leftarrow \mathsf{FKGen}(\mathsf{crs}, pk, sk, f, \tau_\mathtt{f}) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, pk, \tau_\mathtt{p}, x, w)\end{array} : \mathsf{Extract}(\mathsf{crs}, x, \pi, sk_{f,\tau_\mathtt{f}}) = f(w) \,\middle|\, \mathsf{P}(\tau_\mathtt{p}, \tau_\mathtt{f}) = 1\right] \geq 1 - \mathsf{negl}(\lambda).$$

2. ***Functional knowledge.*** *For any PPT adversary* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{l}\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda),\ (pk,sk) \leftarrow \mathsf{UKGen}(\mathsf{crs}) \\ (\pi, x, f, \tau_\mathtt{f}, sk_{f,\tau_\mathtt{f}}) \leftarrow \mathcal{A}(\mathsf{crs}, pk, sk) \\ \quad s.t.\ (x \in \mathcal{L}_\mathcal{R}) \wedge (f \in \mathbb{F}) \wedge \tau_\mathtt{f} \in (\mathcal{T} \cup \{*\}) \\ \quad\quad \wedge (\mathsf{Verify}(\mathsf{crs}, pk, x, \pi) = 1) \\ \quad\quad \wedge (\mathsf{CheckKey}(\mathsf{crs}, pk, f, \tau_\mathtt{f}, sk_{f,\tau_\mathtt{f}}) = 1) \\ \quad\quad \wedge (\mathsf{P}(\mathsf{Ext}_\tau(\pi), \tau_\mathtt{f}) = 1) \\ y \leftarrow \mathsf{Extract}(\mathsf{crs}, x, \pi, sk_{f,\tau_\mathtt{f}})\end{array} : \begin{array}{c}\exists\ w \in \mathcal{W},\ s.t. \\ ((x,w) \in \mathcal{R}) \\ \wedge (y = f(w))\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

3. ***Adaptive soundness.*** *For any computationally unbounded adversary* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{l}\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ (pk, x, \pi) \leftarrow \mathcal{A}(\mathsf{crs})\end{array} : \begin{array}{c}x \notin \mathcal{L}_\mathcal{R} \\ \wedge\ \mathsf{Verify}(\mathsf{crs}, pk, x, \pi) = 1\end{array}\right] \leq \mathsf{negl}(\lambda).$$

4. ***Zero knowledge.*** *For any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$*, there exists a simulator* $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ *such that*

$$\left|\Pr[\mathsf{ExpReal}^{\mathsf{zk}}_{\mathsf{fNIZK},\mathcal{A},n}(\lambda) = 1] - \Pr[\mathsf{ExpIdeal}^{\mathsf{zk}}_{\mathsf{fNIZK},\mathcal{A},\mathsf{Sim},n}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{ExpReal}^{\mathsf{zk}}_{\mathsf{fNIZK},\mathcal{A},n}(\lambda)$ *and* $\mathsf{ExpIdeal}^{\mathsf{zk}}_{\mathsf{fNIZK},\mathcal{A},\mathsf{Sim},n}(\lambda)$ *are defined in Fig. 1, and* $n = \mathsf{poly}(\lambda)$*.*

Here, we offer some explanations and discussions regarding the above definition.

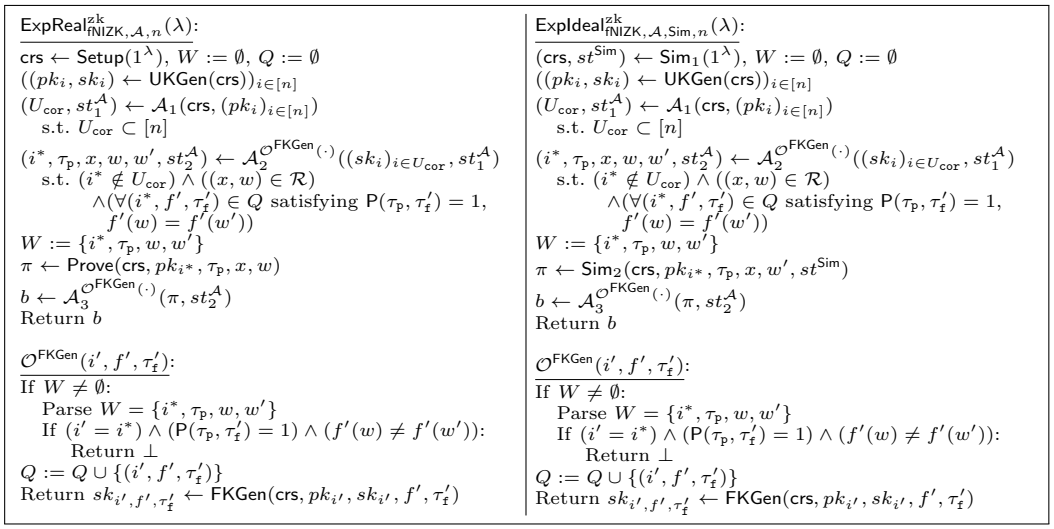

Fig. 1: Games for defining zero knowledge property for fNIZK

1. Completeness of fNIZK has three requirements. Firstly, any normally generated proof π (i.e., π is generated by algorithm Prove) can be verified successfully via algorithm Verify with overwhelming probability. Secondly, for normally generated secret functional key $sk_{f,\tau_{\mathtt{f}}}$ for $(f, \tau_{\mathtt{f}})$, the checking algorithm $\mathsf{CheckKey}(\mathsf{crs}, pk, f, \tau_{\mathtt{f}}, sk_{f,\tau_{\mathtt{f}}})$ returns 1 with overwhelming probability. Thirdly, for normally generated proof π (associated with $\tau_{\mathtt{p}}$) and normally generated secret functional key $sk_{f,\tau_{\mathtt{f}}}$ (for $(f, \tau_{\mathtt{f}})$), if $\mathsf{P}(\tau_{\mathtt{p}}, \tau_{\mathtt{f}}) = 1$, $\mathsf{Extract}$ will extract $f(w)$ with overwhelming probability.
2. The functional knowledge property requires that, in general, if the verifier accepts a proof π, then he/she can be convinced that a function of some witness can be extracted from the proof (with the help of a secret key for the function). Note that this property requires that the secret key should pass the verification of $\mathsf{CheckKey}$, and $\mathsf{P}(\tau_{\mathtt{p}}, \tau_{\mathtt{f}}) = 1$ ($\tau_{\mathtt{p}}$ is the label associated with the proof and $\tau_{\mathtt{f}}$ is the label associated with the secret functional key). Compared with the third requirement of completeness (which focuses on *normally generated proofs*), the functional knowledge property focuses on *maliciously generated proofs which can successfully go through the verification process of* Verify.
3. The security notion of zero knowledge for fNIZK is formalized in the multi-user setting, and requires that except for the fact $(x, w) \in \mathcal{R}$ and the functions about w (i.e., $f'(w)$ for all $(i^*, f', \tau'_{\mathtt{f}}) \in Q$ when $\mathsf{P}(\tau_{\mathtt{p}}, \tau'_{\mathtt{f}}) = 1$), the verifier cannot obtain any other information about w from the proof π. Further explanations regarding the details are provided below. In both $\mathsf{ExpReal}^{\mathsf{zk}}_{\mathsf{fNIZK},\mathcal{A},n}(\lambda)$ and $\mathsf{ExpIdeal}^{\mathsf{zk}}_{\mathsf{fNIZK},\mathcal{A},\mathsf{Sim},n}(\lambda)$,
   (a) $\mathcal{A}$ is allowed to make secret key generation queries to the oracle $\mathcal{O}^{\mathsf{FKGen}}$ adaptively. In particular, we require that for each query $(i', f', \tau'_{\mathtt{f}}) \in [n] \times$

$\Pi.\mathsf{Setup}(1^\lambda)$:	$\Pi.\mathsf{Prove}(\mathsf{crs}^{\mathrm{zk}}, x, w)$:	$\Pi.\mathsf{Verify}(\mathsf{crs}^{\mathrm{zk}}, x, \pi)$:
$\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$	$\tau_{\mathrm{p}} \leftarrow \mathcal{T}$	$b \leftarrow \mathsf{Verify}(\mathsf{crs}, pk, x, \pi)$
$(pk, sk) \leftarrow \mathsf{UKGen}(\mathsf{crs})$	$\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, pk, \tau_{\mathrm{p}}, x, w)$	Return b
Return $\mathsf{crs}^{\mathrm{zk}} = (\mathsf{crs}, pk)$	Return π	

Fig. 2: NIZK Π deduced by fNIZK

$\mathbb{F}\times(\mathcal{T}\cup\{*\})$ raised by $\mathcal{A}_3$ (note that in this case, $W = \{i^*, \tau_{\mathrm{p}}, w, w'\} \neq \emptyset$), $\mathcal{A}_3$ will receive $sk_{i',f',\tau'_{\mathrm{f}}}$ as a response if and only if $(i' \neq i^*) \vee (\mathsf{P}(\tau_{\mathrm{p}}, \tau'_{\mathrm{f}}) = 0) \vee (f'(w) = f'(w'))$. Because if it receives $sk_{i',f',\tau'_{\mathrm{f}}}$ for $(i' = i^*) \wedge (\mathsf{P}(\tau_{\mathrm{p}}, \tau'_{\mathrm{f}}) = 1) \wedge (f'(w) \neq f'(w'))$, it can trivially distinguish the two games.

(b) For the challenge tuple $(i^*, \tau_{\mathrm{p}}, x, w, w')$ output by $\mathcal{A}_2$, we require that (i) $i^* \notin U_{\mathtt{cor}}$, (ii) $(x, w) \in \mathcal{R}$, and (iii) for all $(i^*, f', \tau'_{\mathrm{f}}) \in Q$, it holds that if $\mathsf{P}(\tau_{\mathrm{p}}, \tau'_{\mathrm{f}}) = 1$, then $f'(w) = f'(w')$, where Q denotes all the tuples that have been queried to the oracle $\mathcal{O}^{\mathsf{FKGen}}$ by $\mathcal{A}_2$. We stress that w' is *not* required to be a witness for statement x. The "witness" w', specified by $\mathcal{A}_2$, is used to provide the information (that the simulator Sim is allowed to know) about w to Sim.

We note that the zero knowledge property of fNIZK implies the traditional zero knowledge property. Actually, we have the conclusion that every fNIZK scheme trivially offers a NIZK scheme.

Specifically, for a fNIZK scheme $\mathsf{fNIZK} = (\mathsf{Setup}, \mathsf{UKGen}, \mathsf{FKGen}, \mathsf{CheckKey}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Extract})$, consider a non-interactive proof scheme $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ as in Fig. 2.

We have the following theorem. Due to space limitations, its proof is given in Appendix E.

Theorem 1. *If* fNIZK *is a fNIZK scheme for an* NP *language* $\mathcal{L}_{\mathcal{R}}$*, a function family* $\mathbb{F}$*, a label space* $\mathcal{T}$ *and a predicate function* P*, then* Π *is a NIZK scheme for* $\mathcal{L}_{\mathcal{R}}$*.*

Remark 1. In the definition of zero knowledge for fNIZK, we only consider the single-theorem version. It can be further strengthened to the multi-theorem version, which allows the adversary to generate multiple challenge tuples $(i^*, \tau_{\mathrm{p}}, x, w, w')$. But these tuples should also satisfy $(i^* \notin U_{\mathtt{cor}}) \wedge ((x, w) \in \mathcal{R}) \wedge (\forall (i^*, f', \tau'_{\mathrm{f}}) \in Q$ satisfying $\mathsf{P}(\tau_{\mathrm{p}}, \tau'_{\mathrm{f}}) = 1, f'(w) = f'(w'))$, and meanwhile any secret functional key generation query $(i', f', \tau'_{\mathrm{f}})$ raised by the adversary should satisfy $f'(w) = f'(w')$ or $\mathsf{P}(\tau_{\mathrm{p}}, \tau'_{\mathrm{f}}) = 0$ for all challenge tuples $(i^*, \tau_{\mathrm{p}}, x, w, w')$ (otherwise, the adversary will receive $\perp$ as a response for this query).

4 Generic construction of fNIZK

In this section, we present a method for constructing a fNIZK proof using a functional encryption (FE) scheme and NIZK schemes. The main idea is straightforward: given a valid pair of statement and witness $(x, w) \in \mathcal{R}$, we encrypt the

witness using the functional encryption scheme. Verifiers can then obtain some functions of the witness by decrypting the ciphertext with the corresponding secret keys. In addition to proving the relation about $(x, w) \in \mathcal{R}$, we also need to demonstrate that the ciphertext is well-formed.

We begin by introducing the generic construction, followed by an analysis of its security properties.

Generic construction. Let $\mathcal{L}_{\mathcal{R}}$ be an NP language associated with an NP relation $\mathcal{R} \subset \mathcal{X} \times \mathcal{W}$. Let $\mathbb{F}$ be a function family. Let $\mathcal{T}$ be the label space, and $\mathsf{P} : \mathcal{T} \times (\mathcal{T} \cup \{*\}) \to \{0, 1\}$ be a predicate function satisfying $\mathsf{P}(\tau, *) = 1$ for all $\tau \in \mathcal{T}$.

We define a function family $\widehat{\mathbb{F}}$ as follows: a function $\widehat{f}$ (with domain $\mathcal{W} \times \mathcal{T}$) belongs to $\widehat{\mathbb{F}}$, if and only if there exists $(f, \tau_{\mathtt{f}}) \in \mathbb{F} \times (\mathcal{T} \cup \{*\})$ satisfying

$$\widehat{f}(w, \tau_{\mathtt{p}}) = \begin{cases} f(w) & \text{if } \mathsf{P}(\tau_{\mathtt{p}}, \tau_{\mathtt{f}}) = 1 \\ \perp & \text{if } \mathsf{P}(\tau_{\mathtt{p}}, \tau_{\mathtt{f}}) = 0 \end{cases}$$

For simplicity, for each pair $(f, \tau_{\mathtt{f}}) \in \mathbb{F} \times (\mathcal{T} \cup \{*\})$, we denote the corresponding function in $\widehat{\mathbb{F}}$ as $\widehat{f}_{f,\tau_{\mathtt{f}}}$. We require that there is an efficient algorithm, which takes $(f, \tau_{\mathtt{f}}) \in \mathbb{F} \times (\mathcal{T} \cup \{*\})$ as input and outputs the corresponding $\widehat{f}_{f,\tau_{\mathtt{f}}}$.

Let $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ be a functional encryption scheme for $\widehat{\mathbb{F}}$ on message space $\mathcal{M} = \mathcal{W} \times \mathcal{T}$.

Consider the following two NP relations

$$\mathcal{R}_{\mathrm{ct}} = \{((\tau_{\mathtt{p}}, x, mpk, c), (w, r_{\mathrm{enc}})) : (x, w) \in \mathcal{R} \wedge \mathsf{FE.Enc}(mpk, (w, \tau_{\mathtt{p}}); r_{\mathrm{enc}}) = c\},$$
$$\mathcal{R}_{\mathrm{k}} = \{((mpk, \widehat{f}_{f,\tau_{\mathtt{f}}}, sk_{\widehat{f}_{f,\tau_{\mathtt{f}}}}), (msk, r_{\mathrm{kg}})) : \mathsf{FE.KGen}(mpk, msk, \widehat{f}_{f,\tau_{\mathtt{f}}}; r_{\mathrm{kg}}) = sk_{\widehat{f}_{f,\tau_{\mathtt{f}}}}\},$$

where (x, w) is a statement-witness pair, $\tau_{\mathtt{p}}$ (resp., $\tau_{\mathtt{f}}$) is a label in $\mathcal{T}$ (resp., $\mathcal{T} \cup \{*\}$), mpk and msk is the master key pair of FE, c is a ciphertext, and r_{enc} and r_{kg} are the corresponding randomness. As stated in [16], we can construct NIZKs for any NP language. Therefore, we can construct two NIZK schemes, $\mathsf{NIZK}_{\mathcal{R}_{\mathrm{ct}}} = (\mathsf{NIZK}_{\mathcal{R}_{\mathrm{ct}}}.\mathsf{Setup}, \mathsf{NIZK}_{\mathcal{R}_{\mathrm{ct}}}.\mathsf{Prove}, \mathsf{NIZK}_{\mathcal{R}_{\mathrm{ct}}}.\mathsf{Verify})$ and $\mathsf{NIZK}_{\mathcal{R}_{\mathrm{k}}} = (\mathsf{NIZK}_{\mathcal{R}_{\mathrm{k}}}.\mathsf{Setup}, \mathsf{NIZK}_{\mathcal{R}_{\mathrm{k}}}.\mathsf{Prove}, \mathsf{NIZK}_{\mathcal{R}_{\mathrm{k}}}.\mathsf{Verify})$, for $\mathcal{L}_{\mathcal{R}_{\mathrm{ct}}}$ and $\mathcal{L}_{\mathcal{R}_{\mathrm{k}}}$, respectively.

We present the generic construction of fNIZK proof $\mathsf{fNIZK} = (\mathsf{Setup}, \mathsf{UKGen}, \mathsf{FKGen}, \mathsf{CheckKey}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Extract})$ for $\mathcal{L}_{\mathcal{R}}$, $\mathbb{F}$, $\mathcal{T}$ and P from FE, $\mathsf{NIZK}_{\mathcal{R}_{\mathrm{ct}}}$ and $\mathsf{NIZK}_{\mathcal{R}_{\mathrm{k}}}$, as shown in Fig. 3.

Note that for $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, pk, \tau_{\mathtt{p}}, x, w)$ in Fig. 3, $\tau_{\mathtt{p}}$ is directly packed into π, so an efficient Ext_{τ} can be trivially constructed.

Remark 2. We stress that the above generic scheme fNIZK is built for *any* NP language $\mathcal{L}_{\mathcal{R}}$, *any* function family $\mathbb{F}$ (as long as there is FE for $\widehat{\mathbb{F}}$), and any predicate function $\mathsf{P} : \mathcal{T} \times (\mathcal{T} \cup \{*\}) \to \{0, 1\}$ satisfying $\mathsf{P}(\tau, *) = 1$ for all $\tau \in \mathcal{T}$.

Security analysis. Now, we show that the above fNIZK satisfies completeness, functional knowledge property, adaptive soundness, and zero knowledge.

Completeness. Completeness of fNIZK is trivially guaranteed by correctness of the underlying FE and completeness of the underlying $\mathsf{NIZK}_{\mathcal{R}_{\mathrm{ct}}}$ and $\mathsf{NIZK}_{\mathcal{R}_{\mathrm{k}}}$.

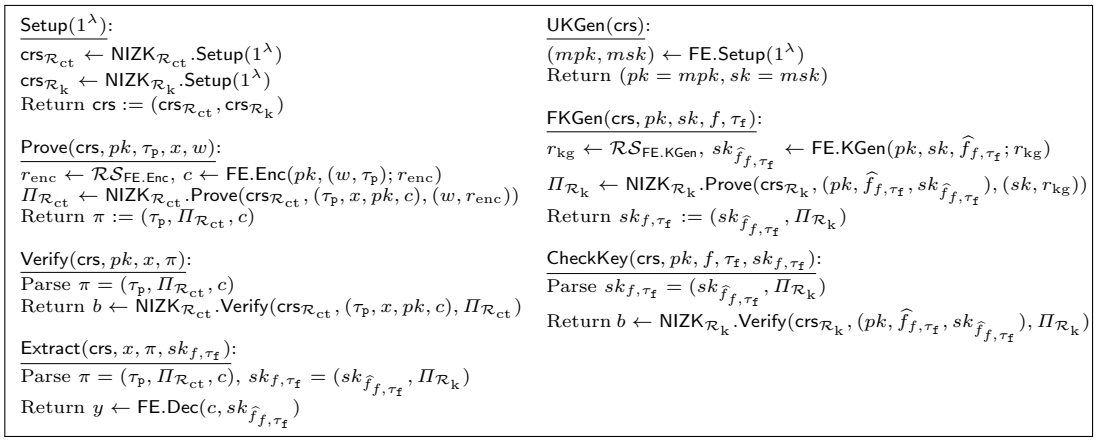

Fig. 3: Construction of fNIZK from FE, $\mathsf{NIZK}_{\mathcal{R}_{\mathrm{ct}}}$ and $\mathsf{NIZK}_{\mathcal{R}_{\mathrm{k}}}$

Functional knowledge. For any PPT adversary $\mathcal{A}$, for $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, $(pk, sk) \leftarrow \mathsf{UKGen}(\mathsf{crs})$, $(\pi, x, f, \tau_{\mathsf{f}}, sk_{f,\tau_{\mathsf{f}}}) \leftarrow \mathcal{A}(\mathsf{crs}, pk, sk)$ satisfying $(x \in \mathcal{L}_{\mathcal{R}}) \wedge (f \in \mathbb{F}) \wedge (\tau_{\mathsf{f}} \in \mathcal{T} \cup \{*\}) \wedge (\mathsf{Verify}(\mathsf{crs}, pk, x, \pi) = 1) \wedge (\mathsf{CheckKey}(\mathsf{crs}, pk, f, \tau_{\mathsf{f}}, sk_{f,\tau_{\mathsf{f}}}) = 1) \wedge (\mathsf{P}(\mathsf{Ext}_\tau(\pi), \tau_{\mathsf{f}}) = 1)$, and for $y \leftarrow \mathsf{Extract}(\mathsf{crs}, x, \pi, sk_{f,\tau_{\mathsf{f}}})$, we analyze the probability that there is w satisfying $((x, w) \in \mathcal{R}) \wedge (y = f(w))$.

First of all, note that $\mathsf{Verify}(\mathsf{crs}, pk, x, \pi) = 1$ implies that $\mathsf{NIZK}_{\mathcal{R}_{\mathrm{ct}}}.\mathsf{Verify}(\mathsf{crs}, (\tau_{\mathsf{p}}, x, pk, c), \Pi_{\mathcal{R}_{\mathrm{ct}}}) = 1$. By the adaptive soundness of $\mathsf{NIZK}_{\mathcal{R}_{\mathrm{ct}}}$, with overwhelming probability, there are $w \in \mathcal{W}$ and $r_{\mathrm{enc}} \in \mathcal{RS}_{\mathsf{FE.Enc}}$ such that $(x, w) \in \mathcal{R}$ and $c = \mathsf{FE.Enc}(mpk, (w, \tau_{\mathsf{p}}); r_{\mathrm{enc}})$, where $\tau_{\mathsf{p}} = \mathsf{Ext}_\tau(\pi)$.

Parse $sk_{f,\tau_{\mathsf{f}}} = (sk_{\widehat{f}_{f,\tau_{\mathsf{f}}}}, \Pi_{\mathcal{R}_{\mathrm{k}}})$. Note that $\mathsf{CheckKey}(\mathsf{crs}, pk, f, \tau_{\mathsf{f}}, sk_{f,\tau_{\mathsf{f}}}) = 1$ implies that $\mathsf{NIZK}_{\mathcal{R}_{\mathrm{k}}}.\mathsf{Verify}(\mathsf{crs}_{\mathcal{R}_{\mathrm{k}}}, (pk, \widehat{f}_{f,\tau_{\mathsf{f}}}, sk_{\widehat{f}_{f,\tau_{\mathsf{f}}}}), \Pi_{\mathcal{R}_{\mathrm{k}}}) = 1$. By the adaptive soundness of $\mathsf{NIZK}_{\mathcal{R}_{\mathrm{k}}}$, with overwhelming probability, $sk_{\widehat{f}_{f,\tau_{\mathsf{f}}}}$ can be explained as generated for $\widehat{f}_{f,\tau_{\mathsf{f}}}$ with $\mathsf{FE.KGen}$. Since $\mathsf{P}(\tau_{\mathsf{p}} = \mathsf{Ext}_\tau(\pi), \tau_{\mathsf{f}}) = 1$, we have $\widehat{f}_{f,\tau_{\mathsf{f}}}(w, \tau_{\mathsf{p}}) = f(w)$. Recall that the algorithm $\mathsf{Extract}$ returns $y \leftarrow \mathsf{FE.Dec}(c, sk_{\widehat{f}_{f,\tau_{\mathsf{f}}}})$, so $y = \widehat{f}_{f,\tau_{\mathsf{f}}}(w, \tau_{\mathsf{p}}) = f(w)$.

Hence, the above fNIZK satisfies the functional knowledge property.

Adaptive soundness. For any computationally unbounded adversary $\mathcal{A}$, let $\mu(\lambda)$ denote the probability that $\mathcal{A}$ outputs pk, $x \notin \mathcal{L}_{\mathcal{R}}$ and $\pi^* = (\tau_{\mathsf{p}}, \Pi^*_{\mathcal{R}_{\mathrm{ct}}}, c)$ such that π^* is an accepting proof, i.e.,

$$\mu(\lambda) = \Pr\left[\begin{array}{l|c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) & \\ (pk, x, \pi^*) \leftarrow \mathcal{A}(\mathsf{crs}) & : \begin{array}{c} x \notin \mathcal{L}_{\mathcal{R}} \\ \wedge\ \mathsf{Verify}(\mathsf{crs}, pk, x, \pi^*) = 1 \end{array} \\ \quad \text{where } \pi^* = (\tau_{\mathsf{p}}, \Pi^*_{\mathcal{R}_{\mathrm{ct}}}, c) & \end{array}\right].$$

Now, we construct an adversary $\mathcal{A}'$, attacking the adaptive soundness of the underlying NIZK $\mathsf{NIZK}_{\mathcal{R}_{\mathrm{ct}}}$, from $\mathcal{A}$ as follows.

Upon receiving the common reference string $\mathsf{crs}_{\mathcal{R}_{\mathrm{ct}}}$, $\mathcal{A}'$ firstly generates $\mathsf{crs}_{\mathcal{R}_{\mathrm{k}}} \leftarrow \mathsf{NIZK}_{\mathcal{R}_{\mathrm{k}}}.\mathsf{Setup}(1^\lambda)$, and then sends $\mathsf{crs} = (\mathsf{crs}_{\mathcal{R}_{\mathrm{ct}}}, \mathsf{crs}_{\mathcal{R}_{\mathrm{k}}})$ to $\mathcal{A}$.

Receiving $(pk, x, \pi^* = (\tau_{\mathrm{p}}, \Pi^*_{\mathcal{R}_{\mathrm{ct}}}, c))$ from $\mathcal{A}$, $\mathcal{A}'$ returns a pair $((\tau_{\mathrm{p}}, x, pk, c), \Pi^*_{\mathcal{R}_{\mathrm{ct}}})$ as its final output.

That is the construction of $\mathcal{A}'$. Next, we analyze its success probability. We have the following equations.

$$
\begin{aligned}
&\Pr\left[\begin{array}{l|c}
\mathsf{crs}_{\mathcal{R}_{\mathrm{ct}}} \leftarrow \mathsf{NIZK}_{\mathcal{R}_{\mathrm{ct}}}.\mathsf{Setup}(1^\lambda) & x^* \notin \mathcal{L}_{\mathcal{R}_{\mathrm{ct}}} \\
(x^*, \Pi^*_{\mathcal{R}_{\mathrm{ct}}}) \leftarrow \mathcal{A}'(\mathsf{crs}_{\mathcal{R}_{\mathrm{ct}}}) & \wedge\ \mathsf{NIZK}_{\mathcal{R}_{\mathrm{ct}}}.\mathsf{Verify}(\mathsf{crs}_{\mathcal{R}_{\mathrm{ct}}}, x^*, \Pi^*_{\mathcal{R}_{\mathrm{ct}}}) = 1 \\
\quad \text{where } x^* = (\tau_{\mathrm{p}}, x, pk, c) &
\end{array}\right] \\
=\ &\Pr\left[\begin{array}{l|c}
\mathsf{crs}_{\mathcal{R}_{\mathrm{ct}}} \leftarrow \mathsf{NIZK}_{\mathcal{R}_{\mathrm{ct}}}.\mathsf{Setup}(1^\lambda) & (\tau_{\mathrm{p}}, x, pk, c) \notin \mathcal{L}_{\mathcal{R}_{\mathrm{ct}}} \\
(pk, x, \pi^*) \leftarrow \mathcal{A}'(\mathsf{crs}_{\mathcal{R}_{\mathrm{ct}}}) & \wedge\ \mathsf{NIZK}_{\mathcal{R}_{\mathrm{ct}}}.\mathsf{Verify}(\mathsf{crs}_{\mathcal{R}_{\mathrm{ct}}}, (\tau_{\mathrm{p}}, x, pk, c), \Pi^*_{\mathcal{R}_{\mathrm{ct}}}) = 1 \\
\quad \text{where } \pi^* = (\tau_{\mathrm{p}}, \Pi^*_{\mathcal{R}_{\mathrm{ct}}}, c) &
\end{array}\right] \\
\geq\ &\Pr\left[\begin{array}{l|c}
\mathsf{crs}_{\mathcal{R}_{\mathrm{ct}}} \leftarrow \mathsf{NIZK}_{\mathcal{R}_{\mathrm{ct}}}.\mathsf{Setup}(1^\lambda) & x \notin \mathcal{L}_{\mathcal{R}} \\
(pk, x, \pi^*) \leftarrow \mathcal{A}'(\mathsf{crs}_{\mathcal{R}_{\mathrm{ct}}}) & \wedge\ \mathsf{NIZK}_{\mathcal{R}_{\mathrm{ct}}}.\mathsf{Verify}(\mathsf{crs}_{\mathcal{R}_{\mathrm{ct}}}, (\tau_{\mathrm{p}}, x, pk, c), \Pi^*_{\mathcal{R}_{\mathrm{ct}}}) = 1 \\
\quad \text{where } \pi^* = (\tau_{\mathrm{p}}, \Pi^*_{\mathcal{R}_{\mathrm{ct}}}, c) &
\end{array}\right] \\
=\ &\Pr\left[\begin{array}{l|c}
\mathsf{crs}_{\mathcal{R}_{\mathrm{ct}}} \leftarrow \mathsf{NIZK}_{\mathcal{R}_{\mathrm{ct}}}.\mathsf{Setup}(1^\lambda) & x \notin \mathcal{L}_{\mathcal{R}} \\
\mathsf{crs}_{\mathcal{R}_{\mathrm{k}}} \leftarrow \mathsf{NIZK}_{\mathcal{R}_{\mathrm{k}}}.\mathsf{Setup}(1^\lambda) & \wedge\ \mathsf{NIZK}_{\mathcal{R}_{\mathrm{ct}}}.\mathsf{Verify}(\mathsf{crs}_{\mathcal{R}_{\mathrm{ct}}}, (\tau_{\mathrm{p}}, x, pk, c), \Pi^*_{\mathcal{R}_{\mathrm{ct}}}) = 1 \\
(pk, x, \pi^*) \leftarrow \mathcal{A}(\mathsf{crs}_{\mathcal{R}_{\mathrm{ct}}}, \mathsf{crs}_{\mathcal{R}_{\mathrm{k}}}) & \\
\quad \text{where } \pi^* = (\tau_{\mathrm{p}}, \Pi^*_{\mathcal{R}_{\mathrm{ct}}}, c) &
\end{array}\right] \\
=\ &\Pr\left[\begin{array}{l|c}
\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) & x \notin \mathcal{L}_{\mathcal{R}} \\
(pk, x, \pi^*) \leftarrow \mathcal{A}(\mathsf{crs}) & \wedge\ \mathsf{Verify}(\mathsf{crs}, pk, x, \pi^*) = 1 \\
\quad \text{where } \pi^* = (\tau_{\mathrm{p}}, \Pi^*_{\mathcal{R}_{\mathrm{ct}}}, c) &
\end{array}\right] \\
=\ &\mu(\lambda).
\end{aligned}
$$

The adaptive soundness of $\mathsf{NIZK}_{\mathcal{R}_{\mathrm{ct}}}$ guarantees that $\mathcal{A}'$'s success probability is negligible, so we derive that $\mu(\lambda) \leq \mathsf{negl}(\lambda)$, concluding the proof of adaptive soundness.

Remark 3. In fact, for any NP language, we can always construct a NIZK in the hidden-bits model [16], which also satisfies adaptive soundness. Therefore, we can always construct $\mathsf{NIZK}_{\mathcal{R}_{\mathrm{ct}}}$ satisfying adaptive soundness.

Zero knowledge. For zero knowledge of fNIZK, we have the following theorem. Due to space limitations, the proof of this theorem is given in Appendix F.

Theorem 2. *If* FE *is IND secure,* $\mathsf{NIZK}_{\mathcal{R}_{\mathrm{ct}}}$ *is single-theorem zero knowledge, and* $\mathsf{NIZK}_{\mathcal{R}_{\mathrm{k}}}$ *is multi-theorem zero knowledge, then* fNIZK *is zero knowledge.*

5 Set membership functional proof

Set membership proofs (SMPs) [5] are widely utilized as building blocks in various cryptographic schemes such as anonymous credentials [7,31], Zcash [29], and e-cash [6]. Following the definition of [5], the relation about set membership is

$$\mathcal{R}_{\mathrm{sm}} = \{((\mathsf{com}, \Phi), (w, r_{\mathrm{com}})) : \mathsf{com} = \mathsf{Com}(pp, w; r_{\mathrm{com}}) \wedge w \in \Phi\},$$

where pp is the public parameter of the commitment scheme in the relation, com is the commitment to the message w, r_{com} denotes the internal randomness, and Φ represents a set. Essentially, given a commitment com, the prover must demonstrate knowledge of the message w corresponding to that commitment, as well as prove that w is an element of the set Φ.

Due to the extensive application value of SMPs, in this section, we provide a concrete and efficient fNIZK for relation about set membership proof. We consider a specific function family as follows: given a public set Φ, each function f within this family corresponds to a subset Φ_{S_f} of Φ, and $f(w)$ indicates whether w belongs to Φ_{S_f} or not. We call fNIZK for $\mathcal{R}_{\mathrm{sm}}$ and the above function family, *set membership functional proof (fSMP)*.

In fSMP, Prove outputs a proof associated with label $\tau_{\mathtt{p}}$ to demonstrate that the committed value $w \in \Phi$, and a verifier with a secret functional key for $(\Phi_{S_f}, \tau_{\mathtt{f}})$ can additionally check whether $w \in \Phi_{S_f}$ or not from the proof, when $\mathsf{P}(\tau_{\mathtt{p}}, \tau_{\mathtt{f}}) = 1$.

To construct a fSMP, we introduce a new primitive, called *set membership encryption (SME)*, and show a generic framework of constructing fSMP from a SME and a commitment scheme that both support Sigma protocols in Sec. 5.1. Subsequently, we propose another primitive called *dual inner-produce encryption (dual IPE)* and illustrate the construction of SME from dual IPE in Sec. 5.2. In Sec. 5.3, we provide an efficient instantiation of dual IPE, utilizing the techniques in IPE [10,33,12], based on the k-LIN assumption. We plug the dual IPE instantiation ($k = 1$) into the generic construction of SME from dual IPE, obtaining a concrete and efficient SME supporting Sigma protocols, in Sec. 5.4. Lastly, we improve the efficiency of fSMP obtained from the concrete SME and Pedersen commitment [27], by utilizing the self-stacking technique [18], in Sec. 5.5.

5.1 fSMP from SME

Here, we firstly introduce set membership encryption (SME) and its security notion, and then show a generic construction of set membership functional proof (fSMP) from SME.

Set membership encryption. Let $\mathcal{W}$ be the message space. We use $\mathcal{S}_{\mathcal{W},l}$ to denote the set of all the sets of size $l = \mathsf{poly}(\lambda)$ in $\mathcal{W}$, i.e., $\mathcal{S}_{\mathcal{W},l} := \{\Phi \subset \mathcal{W} \mid |\Phi| = l\}$. For a set $\Phi \in \mathcal{S}_{\mathcal{W},l}$, without loss of generality, we write that $\Phi = \{w_1, \cdots, w_l\}$. For a set $S \subset [l]$, let $\Phi_S := \{w_j \mid j \in S\}$.

For each set $S \subset [l]$, we define a set membership function $\mathsf{func}_{\Phi_S} : \Phi_S \to \{0,1\}$ as follows: $\mathsf{func}_{\Phi_S}(w) = 1$ if and only if $w \in \Phi_S$.

Definition 2. (Set membership encryption). *Let $\mathcal{T}$ be a label space. Let $\mathsf{P} : \mathcal{T} \times (\mathcal{T} \cup \{*\}) \to \{0,1\}$ be a predicate function satisfying $\mathsf{P}(\tau, *) = 1$ for all $\tau \in \mathcal{T}$. A* set membership encryption (SME) *scheme SME (with set of size $l = \mathsf{poly}(\lambda)$) for message set $\mathcal{W}$, label space $\mathcal{T}$ and predicate P contains five algorithms $\mathsf{SME} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{CheckKey}, \mathsf{Enc}, \mathsf{Query})$.*

- $\mathsf{Setup}(1^\lambda) \to (pk, sk)$*: On input the security parameter* 1^λ*, the setup algorithm outputs a public key* pk *and a secret key* sk*.*
- $\mathsf{KGen}(pk, sk, S, \tau_\mathtt{f}) \to sk_{S,\tau_\mathtt{f}}$*: On input a key pair* (pk, sk)*, a set* $S \subset [l]$ *and a label* $\tau_\mathtt{f} \in \mathcal{T} \cup \{*\}$*, the key generation algorithm outputs a key* $sk_{S,\tau_\mathtt{f}}$*.*
- $\mathsf{CheckKey}(pk, S, \tau_\mathtt{f}, sk_{S,\tau_\mathtt{f}}) \to b$*: On input a public key* pk*, a set* $S \subset [l]$ *and a label* $\tau_\mathtt{f} \in \mathcal{T} \cup \{*\}$ *and a secret key* $sk_{S,\tau_\mathtt{f}}$*, the checking algorithm outputs a bit* $b \in \{0, 1\}$*.*
- $\mathsf{Enc}(pk, \Phi, \tau_\mathtt{p}, w) \to c$*: On input a public key* pk*, a set* $\Phi \subset \mathcal{W}$ *satisfying* $|\Phi| = l$*, a label* $\tau_\mathtt{p} \in \mathcal{T}$*, and a message* $w \in \Phi$*, the encryption algorithm outputs a ciphertext* c*. We assume that there exists an efficient algorithm* Ext_τ *such that* $\tau_\mathtt{p} \leftarrow \mathsf{Ext}_\tau(c)$*.*
- $\mathsf{Query}(c, sk_{S,\tau_\mathtt{f}}) \to y$*: On input a ciphertext* c *and a secret key* $sk_{S,\tau_\mathtt{f}}$*, the query algorithm outputs a bit* $y \in \{0, 1\}$*.*

Correctness requires that for any $\Phi \subset \mathcal{W}$ *satisfying* $|\Phi| = l$*, any* $w \in \Phi$*, any* $S \subset [l]$*, any* $\tau_\mathtt{p} \in \mathcal{T}$ *and any* $\tau_\mathtt{f} \in \mathcal{T} \cup \{*\}$ *satisfying* $\mathsf{P}(\tau_\mathtt{p}, \tau_\mathtt{f}) = 1$*,*

$$\Pr\left[\begin{array}{l}(pk, msk) \leftarrow \mathsf{Setup}(1^\lambda) \\ sk_{S,\tau_\mathtt{f}} \leftarrow \mathsf{KGen}(pk, sk, S, \tau_\mathtt{f}) \\ c \leftarrow \mathsf{Enc}(pk, \Phi, \tau_\mathtt{p}, w), y \leftarrow \mathsf{Query}(c, sk_{S,\tau_\mathtt{f}})\end{array} : y = 1\right] = \begin{cases} 1 & \text{if } w \in \Phi_S \\ \mathsf{negl}(\lambda) & \text{otherwise} \end{cases}$$

$$\Pr\left[\begin{array}{l}(pk, sk) \leftarrow \mathsf{Setup}(1^\lambda) \\ sk_{S,\tau_\mathtt{f}} \leftarrow \mathsf{KGen}(pk, sk, S, \tau_\mathtt{f})\end{array} : \mathsf{CheckKey}(pk, S, \tau_\mathtt{f}, sk_{S,\tau_\mathtt{f}}) = 1\right] \geq 1 - \mathsf{negl}(\lambda).$$

We say that a SME scheme SME *supports Sigma protocols*, if there exists an efficient Sigma protocol for the following relation:

$$\mathcal{R}_c = \{((\tau_\mathtt{p}, c, pk, \Phi), (w, r_{\mathrm{enc}})) : c = \mathsf{Enc}(pk, \Phi, \tau_\mathtt{p}, w; r_{\mathrm{enc}})\}.$$

We now define *IND security* for SME.

Definition 3. (IND security for SME). *A SME scheme* $\mathsf{SME} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{CheckKey}, \mathsf{Enc}, \mathsf{Query})$ *(of size* $l = \mathsf{poly}(\lambda)$*) for message set* $\mathcal{W}$*, label space* $\mathcal{T}$ *and predicate* P *is* IND secure*, if for any PPT adversary* $\mathcal{A}$*, the advantage* $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|$ *is negligible, where* $\mathsf{Exp}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}}(\lambda)$ *is defined in Fig. 4.*

fSMP from SME. Now, we construct a fSMP scheme for the following relation:

$$\mathcal{R}_{\mathrm{sm}} := \{((\mathsf{com}, \Phi), (w, r_{\mathrm{com}})) : \mathsf{com} = \mathsf{Com}(pp, w; r_{\mathrm{com}}) \wedge w \in \Phi\}. \tag{1}$$

We require that the commitment scheme in Eq. (1) also supports Sigma protocols. In other words, there exists an efficient Sigma protocol to prove $\mathsf{com} = \mathsf{Com}(pp, w; r_{\mathrm{com}})$ with statement com and witness (w, r_{com}). One example that satisfies our requirements is Pedersen commitment [27], which we can prove by Okamoto's Sigma protocol [23].

Let $\mathcal{T}$ be the label space. Let $\mathsf{P} : \mathcal{T} \times (\mathcal{T} \cup \{*\}) \to \{0, 1\}$ be a predicate function satisfying $\mathsf{P}(\tau, *) = 1$ for all $\tau \in \mathcal{T}$. Let $\mathsf{SME} = (\mathsf{SME.Setup}, \mathsf{SME.KGen},$

$\underline{\mathsf{Exp}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}}(\lambda)}$:
$(pk, sk) \leftarrow \mathsf{Setup}(1^\lambda)$, $b \leftarrow \{0,1\}$
$W := \emptyset$, $Q := \emptyset$
$(\Phi, \tau_{\mathtt{p}}, w_0, w_1, st) \leftarrow \mathcal{A}_1^{\mathcal{O}^{\mathsf{KGen}}(\cdot)}(pk)$
  s.t. $(w_0 \in \Phi) \wedge (w_1 \in \Phi)$
    $\wedge(\forall(S', \tau'_{\mathtt{f}}) \in Q$ s.t. $\mathsf{P}(\tau_{\mathtt{p}}, \tau'_{\mathtt{f}}) = 1$,
      $\mathsf{func}_{\Phi_{S'}}(w_0) = \mathsf{func}_{\Phi_{S'}}(w_1))$
$W := \{\tau_{\mathtt{p}}, w_0, w_1\}$, $c \leftarrow \mathsf{Enc}(pk, \Phi, \tau_{\mathtt{p}}, w_b)$
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}^{\mathsf{KGen}}(\cdot)}(c, st)$
Return $(b = b')$

$\underline{\mathcal{O}^{\mathsf{KGen}}(S', \tau'_{\mathtt{f}})}$:
If $W \neq \emptyset$:
  Parse $W = \{\tau_{\mathtt{p}}, w_0, w_1\}$
  If $(\mathsf{P}(\tau_{\mathtt{p}}, \tau'_{\mathtt{f}}) = 1) \wedge (\mathsf{func}_{\Phi_{S'}}(w_0) \neq \mathsf{func}_{\Phi_{S'}}(w_1))$:
    Return $\perp$
$Q := Q \cup \{(S', \tau'_{\mathtt{f}})\}$
Return $sk_{S',\tau'_{\mathtt{f}}} \leftarrow \mathsf{KGen}(pk, sk, S', \tau'_{\mathtt{f}})$

Fig. 4: Game for defining IND security for SME

$\mathsf{SME.CheckKey}, \mathsf{SME.Enc}, \mathsf{SME.Query}$) be a SME scheme (with set of size l) for message set $\mathcal{W}$, label space $\mathcal{T}$ and predicate P supporting Sigma protocols. Let $\mathsf{Commit} = (\mathsf{Commit.Setup}, \mathsf{Commit.Com}, \mathsf{Commit.Dec})$ be a commitment scheme supporting Sigma protocols.

Since both SME and Commit support Sigma protocols, a Sigma protocol for relation

$$\widetilde{\mathcal{R}_{\mathrm{sm}}} = \{((\tau_{\mathtt{p}}, \mathsf{com}, c, pk, \Phi), (w, r_{\mathrm{com}}, r_{\mathrm{enc}})) : \mathsf{com} = \mathsf{Commit.Com}(pp, w; r_{\mathrm{com}}) \\ \wedge\, c = \mathsf{SME.Enc}(pk, \Phi, \tau_{\mathtt{p}}, w; r_{\mathrm{enc}})\}$$

can be constructed by the composition of Sigma protocols [3]. So a NIZK scheme $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ (with adaptive soundness) can be obtained by applying the Fiat-Shamir transform [17] to the composite Sigma protocol.

We define the function family $\mathbb{F}$ as follows[6]. Each function $f \in \mathbb{F}$ indicates a set $S_f \subset [l]$, such that for any $x = (\mathsf{com}, \Phi = \{w_1, \cdots, w_l\})$ and any $w_x = (w, r_{\mathrm{com}})$,

$$f(w) = \begin{cases} 1 & \text{if } w \in \Phi_{S_f} \\ 0 & \text{if } w \in \Phi \setminus \Phi_{S_f} \\ \perp & \text{otherwise} \end{cases}$$

We require that there is an efficient algorithm, which takes a $f \in \mathbb{F}$ as input and outputs the corresponding $S_f \subset [l]$. Note that for all $(x, w_x = (w, r_{\mathrm{com}})) \in \mathcal{R}_{\mathrm{sm}}$, we have $f(w) = \mathsf{func}_{\Phi_{S_f}}(w) \in \{0,1\}$.

We present a fSMP scheme $\mathsf{fSMP} = (\mathsf{Setup}, \mathsf{UKGen}, \mathsf{FKGen}, \mathsf{CheckKey}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Extract})$ for $\mathcal{L}_{\mathcal{R}_{\mathrm{sm}}}$, $\mathbb{F}$, $\mathcal{T}$ and P as shown in Fig. 5.

Note that for $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, pk, \tau_{\mathtt{p}}, x, w)$ in Fig. 5, $\tau_{\mathtt{p}}$ is directly packed into π, so an efficient Ext_τ can be trivially constructed.

[6] For each $x \in \mathcal{L}_{\mathcal{R}_{\mathrm{sm}}}$, its witness is in the form of (w, r_{com}). In fSMP, we are only interested in functions of w (rather than r_{com}). So we define $\mathbb{F}$ as family of functions whose domain is $\mathcal{W}$ (rather than $\mathcal{W} \times \mathcal{RS}_{\mathsf{Commit.Com}}$)

$\underline{\mathsf{Setup}(1^\lambda)}$:
$\mathsf{crs}_{\mathrm{nizk}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$
$pp \leftarrow \mathsf{Commit.Setup}(1^\lambda)$
Return $\mathsf{crs} := (\mathsf{crs}_{\mathrm{nizk}}, pp)$

$\underline{\mathsf{UKGen}(\mathsf{crs})}$:
$(pk, sk) \leftarrow \mathsf{SME.Setup}(1^\lambda)$
Return (pk, sk)

$\underline{\mathsf{FKGen}(\mathsf{crs}, pk, sk, f, \tau_{\mathtt{f}})}$:
$sk_{S_f,\tau_{\mathtt{f}}} \leftarrow \mathsf{SME.KGen}(pk, sk, S_f, \tau_{\mathtt{f}})$ $/\!/ S_f \subset [l]$
Return $sk_{f,\tau_{\mathtt{f}}} := sk_{S_f,\tau_{\mathtt{f}}}$

$\underline{\mathsf{CheckKey}(\mathsf{crs}, pk, f, \tau_{\mathtt{f}}, sk_{f,\tau_{\mathtt{f}}})}$:
$b \leftarrow \mathsf{SME.CheckKey}(pk, S_f, \tau_{\mathtt{f}}, sk_{f,\tau_{\mathtt{f}}})$
Return b

$\underline{\mathsf{Prove}(\mathsf{crs}, pk, \tau_{\mathtt{p}}, x, w_x)}$:
Parse $x = (\mathsf{com}, \Phi)$, $w_x = (w, r_{\mathrm{com}})$ $/\!/ |\Phi| = l$
$r_{\mathrm{enc}} \leftarrow \mathcal{RS}_{\mathsf{SME.Enc}}$
$c \leftarrow \mathsf{SME.Enc}(pk, \Phi, \tau_{\mathtt{p}}, w; r_{\mathrm{enc}})$
$\pi_{\widetilde{\mathcal{R}_{\mathrm{sm}}}} \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathrm{nizk}}, (\tau_{\mathtt{p}}, \mathsf{com}, c, pk, \Phi), (w, r_{\mathrm{com}}, r_{\mathrm{enc}}))$
Return $\pi := (\tau_{\mathtt{p}}, \pi_{\widetilde{\mathcal{R}_{\mathrm{sm}}}}, c)$

$\underline{\mathsf{Verify}(\mathsf{crs}, pk, x, \pi)}$:
Parse $x = (\mathsf{com}, \Phi)$, $\pi = (\tau_{\mathtt{p}}, \pi_{\widetilde{\mathcal{R}_{\mathrm{sm}}}}, c)$
$b \leftarrow \mathsf{NIZK.Verify}(\mathsf{crs}_{\mathrm{nizk}}, (\tau_{\mathtt{p}}, \mathsf{com}, c, pk, \Phi), \pi_{\widetilde{\mathcal{R}_{\mathrm{sm}}}})$
Return b

$\underline{\mathsf{Extract}(\mathsf{crs}, x, \pi, sk_{f,\tau_{\mathtt{f}}})}$:
Parse $x = (\mathsf{com}, \Phi)$, $\pi = (\tau_{\mathtt{p}}, \pi_{\widetilde{\mathcal{R}_{\mathrm{sm}}}}, c)$
Return $y \leftarrow \mathsf{SME.Query}(c, sk_{f,\tau_{\mathtt{f}}})$

Fig. 5: Construction of fSMP from SME

Security analysis. Now, we show that the fSMP satisfies completeness, functional knowledge property, adaptive soundness, and zero knowledge.

Completeness. The completeness of fSMP is trivially guaranteed by the completeness of the underlying NIZK and the correctness of the underlying SME.

Functional knowledge. For any $x \in \mathcal{L}_{\mathcal{R}_{\mathrm{sm}}}$ and any $f \in \mathbb{F}$, let $\mathcal{W}_x := \{w \mid \exists\, r_{\mathrm{com}} \text{ s.t. } (x, (w, r_{\mathrm{com}})) \in \mathcal{R}_{\mathrm{sm}}\}$ and $\mathsf{Rge}_{f(\mathcal{W}_x)} := \{f(w) \mid w \in \mathcal{W}_x\}$.

For any PPT adversary $\mathcal{A}$, for $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, $(pk, sk) \leftarrow \mathsf{UKGen}(\mathsf{crs})$, and $(\pi, x, f, \tau_{\mathtt{f}}, sk_{f,\tau_{\mathtt{f}}}) \leftarrow \mathcal{A}(\mathsf{crs}, pk, sk)$ satisfying $(x \in \mathcal{L}_{\mathcal{R}_{\mathrm{sm}}}) \wedge (f \in \mathbb{F}) \wedge (\tau_{\mathtt{f}} \in \mathcal{T} \cup \{*\}) \wedge (\mathsf{Verify}(\mathsf{crs}, pk, x, \pi) = 1) \wedge (\mathsf{CheckKey}(\mathsf{crs}, pk, f, \tau_{\mathtt{f}}, sk_{f,\tau_{\mathtt{f}}}) = 1) \wedge (\mathsf{P}(\tau_{\mathtt{p}}, \tau_{\mathtt{f}}) = 1)$ where $\tau_{\mathtt{p}} \leftarrow \mathsf{Ext}_\tau(\pi)$, we analyze the probability $\Pr[\mathsf{Extract}(\mathsf{crs}, x, \pi, sk_{f,\tau_{\mathtt{f}}}) \notin \mathsf{Rge}_{f(\mathcal{W}_x)}]$ as follows.

First of all, the fact $x \in \mathcal{L}_{\mathcal{R}_{\mathrm{sm}}}$ implies that parsing $x = (\mathsf{com}, \Phi)$, there must be some $w_x = (w, r_{\mathrm{com}})$ satisfying $\mathsf{com} = \mathsf{Commit.Com}(pp, w; r_{\mathrm{com}})$ and $w \in \Phi$. In other words, $\mathcal{W}_x \neq \emptyset$.

Recall that $f \in \mathbb{F}$ indicates a set $S_f \subset [l]$, such that $f(w) = 1$ if $w \in \Phi_{S_f}$, and $f(w) = 0$ if $w \in \Phi \setminus \Phi_{S_f}$.

Parse $\mathsf{crs} = (\mathsf{crs}_{\mathrm{nizk}}, pp)$ and $\pi = (\tau_{\mathtt{p}}, \pi_{\widetilde{\mathcal{R}_{\mathrm{sm}}}}, c)$. Let evt_1 denote the event that $c = \mathsf{SME.Enc}(pk, \Phi, \tau_{\mathtt{p}}, w; r_{\mathrm{enc}})$ for some $w \in \mathcal{W}_x$ and some r_{enc}, and evt_2 denote the event that $c \neq \mathsf{SME.Enc}(pk, \Phi, \tau_{\mathtt{p}}, w; r_{\mathrm{enc}})$ for all $w \in \mathcal{W}_x$ and all r_{enc}. So

$$
\begin{aligned}
&\Pr[\mathsf{Extract}(\mathsf{crs}, x, \pi, sk_{f,\tau_{\mathtt{f}}}) \notin \mathsf{Rge}_{f(\mathcal{W}_x)}] \\
=\ &\Pr[(\mathsf{Extract}(\mathsf{crs}, x, \pi, sk_{f,\tau_{\mathtt{f}}}) \notin \mathsf{Rge}_{f(\mathcal{W}_x)}) \wedge \mathsf{evt}_1] \\
&\quad + \Pr[(\mathsf{Extract}(\mathsf{crs}, x, \pi, sk_{f,\tau_{\mathtt{f}}}) \notin \mathsf{Rge}_{f(\mathcal{W}_x)}) \wedge \mathsf{evt}_2] \\
\leq\ &\Pr[\mathsf{Extract}(\mathsf{crs}, x, \pi, sk_{f,\tau_{\mathtt{f}}}) \notin \mathsf{Rge}_{f(\mathcal{W}_x)} \mid \mathsf{evt}_1] + \Pr[\mathsf{evt}_2].
\end{aligned}
$$

We note that $\mathsf{Extract}(\mathsf{crs}, x, \pi, sk_{f,\tau_\mathsf{f}}) = \mathsf{SME.Query}(c, sk_{f,\tau_\mathsf{f}})$. So if evt_1 occurs (i.e., $c = \mathsf{SME.Enc}(pk, \Phi, \tau_\mathsf{p}, w; r_{\mathrm{enc}})$ for some $w \in \mathcal{W}_x$ and some r_{enc}), the correctness of SME guarantees that $\mathsf{Extract}(\mathsf{crs}, x, \pi, sk_{f,\tau_\mathsf{f}}) = f(w)$ with overwhelming probability. So $\Pr[\mathsf{Extract}(\mathsf{crs}, x, \pi, sk_{f,\tau_\mathsf{f}}) \neq f(w) \mid \mathsf{evt}_1]$ is negligible.

If evt_2 occurs (i.e., $c \neq \mathsf{SME.Enc}(pk, \Phi, \tau_\mathsf{p}, w; r_{\mathrm{enc}})$ for all $w \in \mathcal{W}_x$ and all r_{enc}), then $(\tau_\mathsf{p}, \mathsf{com}, c, pk, \Phi) \notin \mathcal{L}_{\widetilde{\mathcal{R}_{\mathrm{sm}}}}$, where $\mathcal{L}_{\widetilde{\mathcal{R}_{\mathrm{sm}}}}$ is denoted as the the NP language for the relation $\widetilde{\mathcal{R}_{\mathrm{sm}}}$. Note that $\mathsf{Verify}(\mathsf{crs}, x, \pi) = 1$ implies that $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathrm{nizk}}, (\tau_\mathsf{p}, \mathsf{com}, c, pk, \Phi), \pi_{\widetilde{\mathcal{R}_{\mathrm{sm}}}}) = 1$. The adaptive soundness of NIZK for $\widetilde{\mathcal{R}_{\mathrm{sm}}}$ guarantees that

$$\Pr[((\tau_\mathsf{p}, \mathsf{com}, c, pk, \Phi) \notin \mathcal{L}_{\widetilde{\mathcal{R}_{\mathrm{sm}}}}) \wedge (\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathrm{nizk}}, (\tau_\mathsf{p}, \mathsf{com}, c, pk, \Phi), \pi_{\widetilde{\mathcal{R}_{\mathrm{sm}}}}) = 1)]$$

is negligible. So $\Pr[\mathsf{evt}_2]$ is also negligible.

Thus, $\Pr[\mathsf{Extract}(\mathsf{crs}, x, \pi, sk_{f,\tau_\mathsf{f}}) \notin \mathsf{Rge}_{f(\mathcal{W}_x)}]$ is negligible, concluding the proof of functional knowledge.

Adaptive soundness. The adaptive soundness of fSMP is guaranteed by the adaptive soundness of the underlying NIZK.

More specifically, assume that there is a computationally unbounded adversary $\mathcal{A}$, which takes crs as input and outputs (pk, x, π) satisfying $x \notin \mathcal{L}_{\mathcal{R}_{\mathrm{sm}}}$ and $\mathsf{Verify}(\mathsf{crs}, pk, x, \pi) = 1$. Parse $x = (\mathsf{com}, \Phi)$ and $\pi = (\tau_\mathsf{p}, \pi_{\widetilde{\mathcal{R}_{\mathrm{sm}}}}, c)$.

Note that $\mathsf{Verify}(\mathsf{crs}, pk, x, \pi) = 1$ guarantees that $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathrm{nizk}}, (\tau_\mathsf{p}, \mathsf{com}, c, pk, \Phi), \pi_{\widetilde{\mathcal{R}_{\mathrm{sm}}}}) = 1$.

On the other hand, $x \notin \mathcal{L}_{\mathcal{R}_{\mathrm{sm}}}$ implies that there is no $w \in \Phi$ satisfying $\mathsf{com} = \mathsf{Com}(pp, w; \cdot)$. As a result, for all $w \in \Phi$, $(\tau_\mathsf{p}, \mathsf{com}, \mathsf{SME.Enc}(pk, \Phi, \tau_\mathsf{p}, w), pk, \Phi) \notin \mathcal{L}_{\widetilde{\mathcal{R}_{\mathrm{sm}}}}$. In other words, for the $(\tau_\mathsf{p}, \mathsf{com}, pk, \Phi)$ contained in (x, π), and for all ciphertext c', we have $(\tau_\mathsf{p}, \mathsf{com}, c', pk, \Phi) \notin \mathcal{L}_{\widetilde{\mathcal{R}_{\mathrm{sm}}}}$. Hence, for the c contained in π, we derive that $(\tau_\mathsf{p}, \mathsf{com}, c, pk, \Phi) \notin \mathcal{L}_{\widetilde{\mathcal{R}_{\mathrm{sm}}}}$. According to the adaptive soundness of NIZK, this occurs with only negligible probability.

Hence, fSMP is adaptively sound.

Zero knowledge. For zero knowledge of fSMP, we have the following theorem. Due to space limitations, the proof of this theorem is given in Appendix G.

Theorem 3. *If the underlying* SME *is IND secure and supports Sigma protocols, then* fSMP *is zero knowledge.*

5.2 SME from dual IPE

In this part, we introduce a primitive called *dual inner-product encryption (dual IPE)* and leverage it to build a SME scheme.

Dual IPE. For vectors $(\mathbf{x}_1, \mathbf{x}_2), (\mathbf{y}_1, \mathbf{y}_2) \in (\mathbb{Z}_p)^{l_1} \times (\mathbb{Z}_p)^{l_2}$, we define a function $\mathtt{DuIP} : (\mathbb{Z}_p)^{l_1} \times (\mathbb{Z}_p)^{l_2} \times (\mathbb{Z}_p)^{l_1} \times (\mathbb{Z}_p)^{l_2} \to \{0, 1\}$ as follows:

$$\mathtt{DuIP}((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{y}_1, \mathbf{y}_2)) = \begin{cases} 0 & \text{if } \langle \mathbf{x}_1, \mathbf{y}_1 \rangle = \langle \mathbf{x}_2, \mathbf{y}_2 \rangle = 0 \\ 1 & \text{otherwise} \end{cases}$$

Definition 4. (Dual IPE). *A* dual inner-product encryption (dual IPE) *scheme* DIPE *for a message space* $\mathcal{M}$ *and vector space* $(\mathbb{Z}_p)^{l_1} \times (\mathbb{Z}_p)^{l_2}$ *consists of five algorithms* $\mathsf{DIPE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{CheckKey}, \mathsf{Enc}, \mathsf{Dec})$.

- $\mathsf{Setup}(1^\lambda, (l_1, l_2)) \to (pk, msk)$*: On input the security parameter* 1^λ *and the dimension* (l_1, l_2)*, the setup algorithm outputs a public key* pk *and a master secret key* msk*.*
- $\mathsf{KGen}(msk, \mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)) \to sk_{\mathbf{x}}$*: On input a master secret key* msk *and two vectors* $(\mathbf{x}_1, \mathbf{x}_2) \in (\mathbb{Z}_p)^{l_1} \times (\mathbb{Z}_p)^{l_2}$*, the key generation algorithm outputs a secret key* $sk_{\mathbf{x}}$ *for these vectors.*
- $\mathsf{CheckKey}(pk, \mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2), sk_{\mathbf{x}}) \to b$*: On input a public key* pk*, two vectors* $(\mathbf{x}_1, \mathbf{x}_2)$ *and a secret key* $sk_{\mathbf{x}}$*, the checking algorithm outputs a bit* b*.*
- $\mathsf{Enc}(pk, \mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2), m) \to c_{\mathbf{y}}$*: On input* pk*, vectors* $(\mathbf{y}_1, \mathbf{y}_2) \in (\mathbb{Z}_p)^{l_1} \times (\mathbb{Z}_p)^{l_2}$ *and a message* $m \in \mathcal{M}$*, the encryption algorithm outputs a ciphertext* $c_{\mathbf{y}}$ *for* $(\mathbf{y}_1, \mathbf{y}_2)$*.*
- $\mathsf{Dec}(c_{\mathbf{y}}, sk_{\mathbf{x}}) \to m$*: On input a ciphertext* $c_{\mathbf{y}}$ *and a secret key* $sk_{\mathbf{x}}$ *as input, the decryption algorithm outputs a message* m*.*

Correctness requires that for all $m \in \mathcal{M}$ *and all vectors* $(\mathbf{x}_1, \mathbf{x}_2)$ *and* $(\mathbf{y}_1, \mathbf{y}_2)$ *satisfying* $\mathtt{DuIP}((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{y}_1, \mathbf{y}_2)) = 0$*, it holds that:*

$$\Pr\left[\begin{matrix}(pk, msk) \leftarrow \mathsf{Setup}(1^\lambda, (l_1, l_2)) \\ sk_{\mathbf{x}} \leftarrow \mathsf{KGen}(msk, (\mathbf{x}_1, \mathbf{x}_2))\end{matrix} : \mathsf{Dec}(\mathsf{Enc}(pk, (\mathbf{y}_1, \mathbf{y}_2), m), sk_{\mathbf{x}}) = m\right] = 1,$$

$$\Pr\left[\begin{matrix}(pk, msk) \leftarrow \mathsf{Setup}(1^\lambda, (l_1, l_2)) \\ sk_{\mathbf{x}} \leftarrow \mathsf{KGen}(msk, (\mathbf{x}_1, \mathbf{x}_2))\end{matrix} : \mathsf{CheckKey}(pk, (\mathbf{x}_1, \mathbf{x}_2), sk_{\mathbf{x}}) = 1\right] \geq 1 - \mathsf{negl}(\lambda).$$

We now define adaptive security for dual IPE.

Definition 5. (Adaptive security). *A dual IPE scheme* $\mathsf{DIPE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{CheckKey}, \mathsf{Enc}, \mathsf{Dec})$ *for message space* $\mathcal{M}$ *and vector space* $(\mathbb{Z}_p)^{l_1} \times (\mathbb{Z}_p)^{l_2}$ *is* adaptively secure*, if for any PPT adversary* $\mathcal{A}$*, the advantage* $\mathbf{Adv}^{\mathrm{as}}_{\mathsf{DIPE},\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathrm{as}}_{\mathsf{DIPE},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|$ *is negligible, where* $\mathsf{Exp}^{\mathrm{as}}_{\mathsf{DIPE},\mathcal{A}}(\lambda)$ *is defined in Fig. 6.*

The adaptive security implies payload-hiding and partial attribute-hiding. More exactly, our dual IPE requires the first-level vector (i.e., $\mathbf{y}_1^{(\beta)}$) to be fully attribute-hiding, without requiring the second-level vector (i.e., $\mathbf{y}_2$) to be hidden. A concrete adaptively secure dual IPE scheme will be presented in Sec. 5.3.

Encoding algorithms. Before we show how to construct SME from dual IPE, we firstly present two encoding algorithms, as shown in the following:

1. $\mathsf{EncodeW}(\Phi, w)$: Taking $\Phi = \{w_1, \ldots, w_l\}$ and $w \in \mathcal{W}$ as input, it outputs a vector I in $\{0,1\}^l$ as follows:

$$\forall j \in [l] : I_j = \begin{cases} 1 & \text{if } w = w_j \\ 0 & \text{otherwise} \end{cases}$$

For example, supposing $w = w_{j'}$ for some $j' \in [l]$, I can be represented as follows:

$\underline{\mathsf{Exp}^{\mathrm{as}}_{\mathsf{DIPE},\mathcal{A}}(\lambda)}$:
$(pk, msk) \leftarrow \mathsf{Setup}(1^\lambda, (l_1, l_2))$, $b \leftarrow \{0,1\}$, $C_{\mathrm{y,m}} := \emptyset$, $U_{\mathrm{x}} := \emptyset$
$(\mathbf{y}_1^{(0)}, \mathbf{y}_1^{(1)}, \mathbf{y}_2, m_0, m_1, st) \leftarrow \mathcal{A}_1^{\mathcal{O}^{\mathsf{KGen}}(\cdot)}(pk, (l_1, l_2))$
s.t. if $m_0 \neq m_1$, $\mathtt{DuIP}((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{y}_1^{(\beta)}, \mathbf{y}_2)) \neq 0$ for all $\beta \in \{0,1\}$ and all $(\mathbf{x}_1, \mathbf{x}_2) \in U_{\mathrm{x}}$;
if $m_0 = m_1$, $(\forall \beta \in \{0,1\} : \mathtt{DuIP}((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{y}_1^{(\beta)}, \mathbf{y}_2)) \neq 0)$
$\vee (\forall \beta \in \{0,1\} : \mathtt{DuIP}((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{y}_1^{(\beta)}, \mathbf{y}_2)) = 0)$ for all $(\mathbf{x}_1, \mathbf{x}_2) \in U_{\mathrm{x}}$
$C_{\mathrm{y,m}} := \{(\mathbf{y}_1^{(0)}, \mathbf{y}_1^{(1)}, \mathbf{y}_2, m_0, m_1)\}$, $c \leftarrow \mathsf{Enc}(pk, (\mathbf{y}_1^{(b)}, \mathbf{y}_2), m_b)$, $b' \leftarrow \mathcal{A}_2^{\mathcal{O}^{\mathsf{KGen}}(\cdot)}(c, st)$
Return $(b = b')$

$\underline{\mathcal{O}^{\mathsf{KGen}}((\mathbf{x}_1, \mathbf{x}_2))}$:
If $C_{\mathrm{y,m}} \neq \emptyset$:
Parse $C_{\mathrm{y,m}} = \{(\mathbf{y}_1^{(0)}, \mathbf{y}_1^{(1)}, \mathbf{y}_2, m_0, m_1)\}$
If $m_0 \neq m_1$:
If $(\exists \beta \in \{0,1\} : \mathtt{DuIP}((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{y}_1^{(\beta)}, \mathbf{y}_2)) = 0)$: Return $\perp$
If $m_0 = m_1$:
If $(\exists \beta \in \{0,1\} : (\mathtt{DuIP}((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{y}_1^{(\beta)}, \mathbf{y}_2)) = 0) \wedge (\mathtt{DuIP}((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{y}_1^{(1-\beta)}, \mathbf{y}_2)) \neq 0))$:
Return $\perp$
$U_{\mathrm{x}} := U_{\mathrm{x}} \cup \{(\mathbf{x}_1, \mathbf{x}_2)\}$
Return $sk_{\mathbf{x}} \leftarrow \mathsf{KGen}(msk, (\mathbf{x}_1, \mathbf{x}_2))$

Fig. 6: Game for defining adaptive security for dual IPE

$$I = \underset{1}{\boxed{0}}\boxed{\cdots}\boxed{0}\underset{j'}{\boxed{1}}\boxed{0}\boxed{\cdots}\underset{l}{\boxed{0}}$$

It is clear that I depends on w. Thus, we also write $I(w)$ for simplicity to denote the vector output by $\mathsf{EncodeW}(\Phi, w)$.

2. $\mathsf{EncodeS}(l, S)$: Taking a value l and a set $S \subset [l]$ as input, it outputs a vector I^* in $\{0,1\}^l$ as follows:

$$\forall j \in [l] : I^*_j = \begin{cases} 0 & \text{if } j \in S \\ 1 & \text{otherwise} \end{cases}$$

For example, supposing $S = \{3, 5, l\} \subset [l]$, I^* can be represented as follows:

$$I^* = \underset{1}{\boxed{1}}\underset{3}{\boxed{1}}\boxed{0}\underset{5}{\boxed{1}}\boxed{0}\boxed{1}\boxed{\cdots}\boxed{1}\underset{l}{\boxed{0}}$$

It is clear that given l, I^* depends on S. Thus, we also write $I^*(S)$ for simplicity to denote the vector output by $\mathsf{EncodeS}(l, S)$ when l is given in the text.

SME from dual IPE. Let $\mathcal{I} = \{0,1\}^{l_1}$ denote a vector space with alphabet $\{0,1\}$ and length l_1. Let $\mathcal{T} = (\mathbb{Z}_p)^{l_2}$ denote the label space with alphabet

$\{0,\ldots,p-1\}$ and length l_2. Let $\mathsf{P}_{\mathrm{ip}} : \mathcal{T} \times (\mathcal{T} \cup \{*\}) \to \{0,1\}$ be a predicate function satisfying that for any $\tau \in \mathcal{T}$ and $\tau' \in \mathcal{T} \cup \{*\}$,

$$\mathsf{P}_{\mathrm{ip}}(\tau,\tau') = \begin{cases} 1 & \text{if } (\tau' = *) \vee ((\tau' \neq *) \wedge (\langle \tau, \tau' \rangle = 0)) \\ 0 & \text{otherwise} \end{cases}$$

From the perspective of inner product, the symbol $*$ can be regarded as vector 0^{l_2}. In this case, P_{ip} can be rephrased as

$$\mathsf{P}_{\mathrm{ip}}(\tau,\tau') = \begin{cases} 1 & \text{if } \langle \tau, \tau' \rangle = 0 \\ 0 & \text{otherwise} \end{cases}$$

Let $\mathsf{DIPE} = (\mathsf{DIPE.Setup}, \mathsf{DIPE.KGen}, \mathsf{DIPE.CheckKey}, \mathsf{DIPE.Enc}, \mathsf{DIPE.Dec})$ be an adaptively secure dual IPE scheme for vector space $\mathcal{I} \times \mathcal{T}$ and message space $\mathcal{M}_{\mathsf{DIPE}}$, as presented in Sec. 5.3. Let $\mathsf{m}_{\mathtt{dum}}$ be an arbitrary public default message in $\mathcal{M}_{\mathsf{DIPE}}$.

Our SME scheme $\mathsf{SME} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{CheckKey}, \mathsf{Enc}, \mathsf{Query})$ (with set of size l_1) for message set $\mathcal{W}$, label space $\mathcal{T}$ and predicate P_{ip} is described in Fig. 7.

$\underline{\mathsf{Setup}(1^\lambda)}$: //set $l_1 = \mathsf{poly}(\lambda)$, $l_2 = \mathsf{poly}(\lambda)$
$(pk, msk) \leftarrow \mathsf{DIPE.Setup}(1^\lambda, (l_1, l_2))$
Return $(pk, sk = msk)$

$\underline{\mathsf{KGen}(pk, sk, S, \tau_{\mathtt{f}})}$: // $S \subset [l_1]$
$I^*(S) \leftarrow \mathsf{EncodeS}(l_1, S)$
$sk_{I^*(S),\tau_{\mathtt{f}}} \leftarrow \mathsf{DIPE.KGen}(msk, (I^*(S), \tau_{\mathtt{f}}))$
Return $sk_{S,\tau_{\mathtt{f}}} = sk_{I^*(S),\tau_{\mathtt{f}}}$

$\underline{\mathsf{CheckKey}(pk, S, \tau_{\mathtt{f}}, sk_{S,\tau_{\mathtt{f}}})}$:
Return $b \leftarrow \mathsf{DIPE.CheckKey}(pk, (I^*(S), \tau_{\mathtt{f}}), sk_{S,\tau_{\mathtt{f}}})$

$\underline{\mathsf{Enc}(pk, \Phi, \tau_{\mathtt{p}}, w)}$: // $|\Phi| = l_1$
If $w \notin \Phi$: Return $\perp$
$I(w) = \mathsf{EncodeW}(\Phi, w)$
$c \leftarrow \mathsf{DIPE.Enc}(pk, (I(w), \tau_{\mathtt{p}}), \mathsf{m}_{\mathtt{dum}})$
Return c

$\underline{\mathsf{Query}(c, sk_{S,\tau_{\mathtt{f}}})}$:
$\mathsf{m}'_{\mathtt{dum}} \leftarrow \mathsf{DIPE.Dec}(c, sk_{S,\tau_{\mathtt{f}}})$
If $\mathsf{m}_{\mathtt{dum}} = \mathsf{m}'_{\mathtt{dum}}$: Return $y = 1$
Else: Return $y = 0$

Fig. 7: Construction of SME from DIPE ($\mathsf{m}_{\mathtt{dum}}$ is an arbitrary public default message in $\mathcal{M}_{\mathsf{DIPE}}$)

Security analysis. Here, we show that the above SME constructed from dual IPE DIPE, is correct and IND secure.

Correctness. For any $\Phi \subset \mathcal{W}$ satisfying $|\Phi| = l$, any $w \in \Phi$ and any $S \subset [l]$, any $\tau_{\mathtt{f}} \in \mathcal{T} \cup \{*\}$ and any $\tau_{\mathtt{p}} \in \mathcal{T}$ satisfying $\mathsf{P}_{\mathrm{ip}}(\tau_{\mathtt{p}}, \tau_{\mathtt{f}}) = 1$ (i.e., $\langle \tau_{\mathtt{p}}, \tau_{\mathtt{f}} \rangle = 0$), for $(pk, sk) \leftarrow \mathsf{Setup}(1^\lambda)$, $sk_{S,\tau_{\mathtt{f}}} \leftarrow \mathsf{KGen}(pk, sk, S, \tau_{\mathtt{f}})$, $c \leftarrow \mathsf{Enc}(pk, \Phi, \tau_{\mathtt{p}}, w)$ and $y \leftarrow \mathsf{Query}(c, sk_{S,\tau_{\mathtt{f}}})$, it holds that

- If $w \in \Phi_S$, then $\langle I^*(S), I(w) \rangle = 0$. So $\mathsf{DIPE.Dec}(c, sk_{S,\tau_{\mathtt{f}}}) = \mathsf{m}_{\mathtt{dum}}$, which suggests that $y = 1$.

– If $w \notin \Phi_S$, then $\langle I^*(S), I(w)\rangle \neq 0$. The adaptive security of DIPE guarantee that $\mathsf{DIPE.Dec}(c, sk_{S,\tau_\mathtt{f}}) \neq \mathsf{m}_{\mathtt{dum}}$ with overwhelming probability. Hence, the probability that $y = 1$ is negligible.

For $(pk, sk) \leftarrow \mathsf{Setup}(1^\lambda)$ and $sk_{S,\tau_\mathtt{f}} \leftarrow \mathsf{KGen}(pk, sk, S, \tau_\mathtt{f})$, we have that $\mathsf{CheckKey}(pk, S, \tau_\mathtt{f}, sk_{S,\tau_\mathtt{f}}) = \mathsf{DIPE.CheckKey}(pk, (I^*(S), \tau_\mathtt{f}), sk_{S,\tau_\mathtt{f}})$, since $\mathsf{CheckKey}$ invokes $\mathsf{DIPE.CheckKey}$ to check the keys. By the correctness of DIPE, we know that $\mathsf{DIPE.CheckKey}(pk, (I^*(S), \tau_\mathtt{f}), sk_{S,\tau_\mathtt{f}}) = 1$, when $sk_{S,\tau_\mathtt{f}} \leftarrow \mathsf{DIPE.KGen}(pk, (I^*(S), \tau_\mathtt{f}))$. Note that the key generation KGen calls $\mathsf{DIPE.KGen}(pk, (I^*(S), \tau_\mathtt{f}))$ to generate $sk_{S,\tau_\mathtt{f}}$. Thus, $\mathsf{CheckKey}(pk, S, \tau_\mathtt{f}, sk_{S,\tau_\mathtt{f}}) = \mathsf{DIPE.CheckKey}(pk, (I^*(S), \tau_\mathtt{f}), sk_{S,\tau_\mathtt{f}}) = 1$.

In all, the SME above is correct.

IND security. The IND security of SME is guaranteed by the adaptive security of the underlying dual IPE DIPE. Formally, we have the following theorem. Due to space limitations, its proof will be given in Appendix H.

Theorem 4. *If the underlying* DIPE *is adaptively secure, then* SME *is IND secure.*

5.3 A concrete construction of dual IPE

Starting from the IPE schemes proposed in [10,12,33], we provide a concrete construction of adaptively secure dual IPE. In this subsection, we first give some notations, assumptions and facts. Then, we provide a *private-key* dual IPE, and upgrade it to *public-key* dual IPE employing the "private-key to public-key" compiler in [33]. Unless otherwise specified, the term "dual IPE" refers to public-key dual IPE.

Notations. A group generator $\mathcal{G}$ takes as input the security parameter λ and outputs group description $\mathbb{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, where p is a prime of $\Theta(\lambda)$ bits, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of order p, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a nondegenerate bilinear map. We require that group operations in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ as well the bilinear map e are computable in deterministic polynomial time with respect to λ. Let $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and $g_T = e(g_1, g_2) \in \mathbb{G}_T$ be the respective generators. We employ the implicit representation of group elements: for a matrix $\mathbf{M}$ over $\mathbb{Z}_p$, we define $[\mathbf{M}]_1 = g_1^{\mathbf{M}}$,$[\mathbf{M}]_2 = g_2^{\mathbf{M}}$,$[\mathbf{M}]_T = g_T^{\mathbf{M}}$, where exponentiations are carried out component-wise. For $[\mathbf{A}]_1$ and $[\mathbf{B}]_2$, we let $e([\mathbf{A}]_1, [\mathbf{B}]_2) = [\mathbf{AB}]_T$. For any matrix $\mathbf{B} \in \mathbb{Z}_p^{n\times m}$, we define an operator $\odot$ as follows: $\alpha \odot [\mathbf{B}]_* = [\alpha\mathbf{B}]_*$, where the star $*$ belongs to $\{1, 2, T\}$ and α could be a constant in $\mathbb{Z}_p$, a row vector in $\mathbb{Z}_p^n$ or a matrix in $\mathbb{Z}_p^{n'\times n}$. $\mathrm{GL}_k(\mathbb{Z}_p)$ denotes the general linear group of degree k over $\mathbb{Z}_p$. Let $\mathbf{A}$ be a $\ell \times k$ matrix over $\mathbb{Z}_p$, where $\ell \geq k$. We use $\mathsf{span}_c(\mathbf{A})$ to denote the column span of $\mathbf{A}$. By $\mathsf{span}_r(\mathbf{A}^\top)$, we are indicating the row span of $\mathbf{A}^\top$. If $\mathbf{A}$ is a basis of $\mathsf{span}_c(\mathbf{A})$, we use $\mathsf{basis}(\mathbf{A})$ to denote another basis of $\mathsf{span}_c(\mathbf{A})$ via $\mathbf{A} \cdot \mathbf{R}$, where $\mathbf{R} \leftarrow \mathrm{GL}_k(\mathbb{Z}_p)$. Given an invertible matrix $\mathbf{B}$, we use $\mathbf{B}^*$ to denote its dual satisfying $\mathbf{B}^\top\mathbf{B}^* = \mathbf{I}$.

Assumptions. We review the matrix Diffie-Hellman (MDDH) assumption on $\mathbb{G}_1$ [15]. The $\mathrm{MDDH}_{k,\ell}$ assumption on $\mathbb{G}_2$ can be defined analogously.

Definition 6. ($\mathrm{MDDH}_{k,\ell}$ **assumption).** *Let $\ell > k \geq 1$. We say that the $\mathrm{MDDH}_{k,\ell}$ assumption holds with respect to $\mathcal{G}$ if for all PPT adversaries $\mathcal{A}$, the following advantage function is negligible in λ.*

$$\mathbf{Adv}_{\mathcal{A}}^{\mathrm{MDDH}_{k,\ell}}(\lambda) := |\Pr[\mathcal{A}(\mathbb{G}, [\mathbf{M}]_1, [\mathbf{Ms}]_1) = 1] - \Pr[\mathcal{A}(\mathbb{G}, [\mathbf{M}]_1, [\mathbf{u}]_1) = 1]|$$

where $\mathbb{G} \leftarrow \mathcal{G}(1^\lambda)$, $\mathbf{M} \leftarrow \mathbb{Z}_p^{\ell \times k}$, $\mathbf{s} \leftarrow \mathbb{Z}_p^k$ and $\mathbf{u} \leftarrow \mathbb{Z}_p^\ell$.

Let $\ell_1, \ell_2, \ell_3 > 1$ and $\ell := \ell_1 + \ell_2 + \ell_3$. We use basis $\mathbf{B}_1 \leftarrow \mathbb{Z}_p^{\ell \times \ell_1}$, $\mathbf{B}_2 \leftarrow \mathbb{Z}_p^{\ell \times \ell_2}$, $\mathbf{B}_3 \leftarrow \mathbb{Z}_p^{\ell \times \ell_3}$, and its dual basis $(\mathbf{B}_1^*, \mathbf{B}_2^*, \mathbf{B}_3^*)$ such that $\mathbf{B}_i^\top \mathbf{B}_i^* = \mathbf{I}$ (known as *non-degeneracy*) and $\mathbf{B}_i^\top \mathbf{B}_j^* = 0$ if $i \neq j$ (known as *orthogonality*). We review the $\mathrm{SD}^{\mathbb{G}_2}_{\mathbf{B}_1 \mapsto \mathbf{B}_1, \mathbf{B}_2}$ assumption as follows. By symmetry, one may permute the indices for subspaces.

Definition 7. ($\mathrm{SD}^{\mathbb{G}_2}_{\mathbf{B}_1 \mapsto \mathbf{B}_1, \mathbf{B}_2}$ **assumption).** *We say that the $\mathrm{SD}^{\mathbb{G}_2}_{\mathbf{B}_1 \mapsto \mathbf{B}_1, \mathbf{B}_2}$ assumption holds if for all PPT adversaries $\mathcal{A}$, the following advantage function is negligible in λ.*

$$\mathbf{Adv}_{\mathcal{A}}^{\mathrm{SD}^{\mathbb{G}_2}_{\mathbf{B}_1 \mapsto \mathbf{B}_1, \mathbf{B}_2}}(\lambda) := |\Pr[\mathcal{A}(\mathbb{G}, D, [\mathbf{t}_0]_1) = 1] - \Pr[\mathcal{A}(\mathbb{G}, D, [\mathbf{t}_1]_1) = 1]|$$

where $D := ([\mathbf{B}_1]_2, [\mathbf{B}_2]_2, [\mathbf{B}_3]_2, \mathsf{basis}(\mathbf{B}_1^*, \mathbf{B}_2^*), \mathsf{basis}(\mathbf{B}_3^*))$, $\mathbf{t}_0 \leftarrow \mathsf{span}_c(\mathbf{B}_1)$, $\mathbf{t}_1 \leftarrow \mathsf{span}_c(\mathbf{B}_1, \mathbf{B}_2)$.

It is known that k-LIN $\Rightarrow \mathrm{MDDH}_{k,\ell}$ [15] and $\mathrm{MDDH}_{\ell,\ell_1+\ell_2} \Rightarrow \mathrm{SD}^{\mathbb{G}_2}_{\mathbf{B}_1 \mapsto \mathbf{B}_1, \mathbf{B}_2}$ [11].

Facts. With basis $(\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3)$, we can uniquely decompose $\mathbf{w} \in \mathbb{Z}_p^{1 \times \ell}$ as $\mathbf{w} = \Sigma_{\beta \in [3]} \mathbf{w}^{(\beta)}$ where $\mathbf{w}^{(\beta)} \in \mathsf{span}_r(\mathbf{B}_\beta^{*\top})$. Define $\mathbf{w}^{(\beta_1 \beta_2)} = \mathbf{w}^{(\beta_1)} + \mathbf{w}^{(\beta_2)}$ for $\beta_1, \beta_2 \in [3]$. We have the following two facts:

1. For $\beta \in [3]$, it holds that $\mathbf{w}\mathbf{B}_\beta = \mathbf{w}^{(\beta)}\mathbf{B}_\beta$;
2. For all $\beta^* \in [3]$, it holds that $\{\mathbf{w}^{(\beta^*)}, \{\mathbf{w}^{(\beta)}\}_{\beta \neq \beta^*}\} \equiv \{\mathbf{w}^*, \{\mathbf{w}^{(\beta)}\}_{\beta \neq \beta^*}\}$ when $\mathbf{w} \leftarrow \mathbb{Z}_p^{1 \times \ell}$ and $\mathbf{w}^* \leftarrow \mathsf{span}_r(\mathbf{B}_{\beta^*}^{*\top})$.

Construction of private-key dual IPE. In a private-key dual IPE, the Setup algorithm does not output pk; the $\mathsf{CheckKey}$ algorithm is not needed; and the Enc algorithm takes msk instead of pk as input. The adaptive security can be defined similar to Def. 5 except that the adversary $\mathcal{A}$ only gets the challenge ciphertext c and has access to KGen. Next, we give a concrete construction of private-key dual IPE $\mathsf{skDIPE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$, and the details are shown in Fig. 8.

Correctness. For all vectors $(\mathbf{x}_1, \mathbf{x}_2)$ and $(\mathbf{y}_1, \mathbf{y}_2)$ satisfying $\mathtt{DuIP}((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{y}_1, \mathbf{y}_2)) = 0$, we have

$$\begin{aligned}
&((x_{1,1} \cdot C_{1,1} + \cdots + x_{1,l_1} \cdot C_{1,l_1} + x_{2,1} \cdot C_{2,1} + \cdots + x_{2,l_2} \cdot C_{2,l_2}) \odot K_1) \cdot K_0^{-1} \\
=&[(x_{1,1} \cdot (y_{1,1}\mathbf{u}_1 + \mathbf{w}_{1,1}) + \cdots + x_{1,l_1} \cdot (y_{1,l_1}\mathbf{u}_1 + \mathbf{w}_{1,l_1}) \\
&\quad + x_{2,1} \cdot (y_{2,1}\mathbf{u}_2 + \mathbf{w}_{2,1}) + \cdots + x_{2,l_2} \cdot (y_{2,l_2}\mathbf{u}_2 + \mathbf{w}_{2,l_2}))\mathbf{B}_1\mathbf{r}]_2
\end{aligned}$$

$\underline{\mathsf{Setup}(1^\lambda, l_1, l_2)}$:
$\mathbb{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$
$\mathbf{B}_1 \leftarrow \mathbb{Z}_p^{(2k+1)\times k}$
$\mathbf{u}_1, \mathbf{u}_2, \mathbf{w}_{1,1}, \ldots, \mathbf{w}_{1,l_1}, \mathbf{w}_{2,1}, \ldots, \mathbf{w}_{2,l_2} \leftarrow \mathbb{Z}_p^{1\times(2k+1)}$, $\alpha \leftarrow \mathbb{Z}_p$
$msk := (\mathbb{G}, \alpha, \mathbf{u}_1, \mathbf{u}_2, \mathbf{w}_{1,1}, \ldots, \mathbf{w}_{1,l_1}, \mathbf{w}_{2,1}, \ldots, \mathbf{w}_{2,l_2}, \mathbf{B}_1)$
Return msk

$\underline{\mathsf{KGen}(msk, (\mathbf{x}_1, \mathbf{x}_2))}$:
Parse $\mathbf{x}_1 = (x_{1,1}, \ldots, x_{1,l_1}) \in \mathbb{Z}_p^{l_1}$, $\mathbf{x}_2 = (x_{2,1}, \ldots, x_{2,l_2}) \in \mathbb{Z}_p^{l_2}$
$\mathbf{r} \leftarrow \mathbb{Z}_p^k$
$K_0 := [\alpha + (x_{1,1}\cdot\mathbf{w}_{1,1} + \ldots + x_{1,l_1}\cdot\mathbf{w}_{1,l_1} + x_{2,1}\cdot\mathbf{w}_{2,1} + \ldots + x_{2,l_2}\cdot\mathbf{w}_{2,l_2})\mathbf{B}_1\mathbf{r}]_2$
$K_1 := [\mathbf{B}_1\mathbf{r}]_2$
Return $sk_{\mathbf{x}} := (K_0, K_1)$

$\underline{\mathsf{Enc}(msk, (\mathbf{y}_1, \mathbf{y}_2), m \in \mathbb{G}_2)}$:
Parse $\mathbf{y}_1 = (y_{1,1}, \ldots, y_{1,l_1}) \in \mathbb{Z}_p^{l_1}$, $\mathbf{y}_2 = (y_{2,1}, \ldots, y_{2,l_2}) \in \mathbb{Z}_p^{l_2}$
For $i \in [l_1]$: $C_{1,i} := y_{1,i}\cdot\mathbf{u}_1 + \mathbf{w}_{1,i}$
For $i \in [l_2]$: $C_{2,i} := y_{2,i}\cdot\mathbf{u}_2 + \mathbf{w}_{2,i}$
$C := [\alpha]_2 \cdot m$
Return $c_{\mathbf{y}} := ((C_{1,i})_{i\in[l_1]}, (C_{2,i})_{i\in[l_2]}, C)$

$\underline{\mathsf{Dec}(c_{\mathbf{y}}, sk_{\mathbf{x}})}$:
Parse $c_{\mathbf{y}} = ((C_{1,i})_{i\in[l_1]}, (C_{2,i})_{i\in[l_2]}, C)$, $sk_{\mathbf{x}} = (K_0, K_1)$
Return $m' := C \cdot ((x_{1,1}\cdot C_{1,1} + \cdots + x_{1,l_1}\cdot C_{1,l_1} + x_{2,1}\cdot C_{2,1} + \cdots + x_{2,l_2}\cdot C_{2,l_2}) \odot K_1) \cdot K_0^{-1}$

Fig. 8: The algorithms of private-key dual IPE scheme skDIPE

$$
\begin{aligned}
&\cdot [\alpha + (x_{1,1}\cdot\mathbf{w}_{1,1} + \ldots + x_{1,l_1}\cdot\mathbf{w}_{1,l_1} + x_{2,1}\cdot\mathbf{w}_{2,1} + \ldots + x_{2,l_2}\cdot\mathbf{w}_{2,l_2})\mathbf{B}_1\mathbf{r}]_2^{-1} \\
=&[\langle\mathbf{x}_1, \mathbf{y}_1\rangle \cdot \mathbf{u}_1\mathbf{B}_1\mathbf{r} + \langle\mathbf{x}_2, \mathbf{y}_2\rangle \cdot \mathbf{u}_2\mathbf{B}_1\mathbf{r}]_2 \\
&\cdot [(x_{1,1}\cdot\mathbf{w}_{1,1} + \ldots + x_{1,l_1}\cdot\mathbf{w}_{1,l_1} + x_{2,1}\cdot\mathbf{w}_{2,1} + \ldots + x_{2,l_2}\cdot\mathbf{w}_{2,l_2})\mathbf{B}_1\mathbf{r}]_2 \\
&\cdot [\alpha]_2^{-1} \cdot [(x_{1,1}\cdot\mathbf{w}_{1,1} + \ldots + x_{1,l_1}\cdot\mathbf{w}_{1,l_1} + x_{2,1}\cdot\mathbf{w}_{2,1} + \ldots + x_{2,l_2}\cdot\mathbf{w}_{2,l_2})\mathbf{B}_1\mathbf{r}]_2^{-1} \\
=&[\alpha]_2^{-1}
\end{aligned}
$$

where the last equality follows from the fact that $\langle\mathbf{x}_1, \mathbf{y}_1\rangle = 0$ and $\langle\mathbf{x}_2, \mathbf{y}_2\rangle = 0$. This proves the correctness.

Security. We now state the security theorem of our proposed private-key dual IPE scheme. Due to space limitations, the proof is given in Appendix I.

Theorem 5. *Under the k-LIN assumption, the private-key dual IPE scheme described in Fig. 8 is adaptively secure.*

Construction of public-key dual IPE. We give a concrete construction of public-key dual IPE $\mathsf{pkDIPE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{CheckKey}, \mathsf{Enc}, \mathsf{Dec})$, and the details are shown in Fig. 9.

Correctness. For all $m \in \mathcal{M}$ and all vectors $(\mathbf{x}_1, \mathbf{x}_2)$ and $(\mathbf{y}_1, \mathbf{y}_2)$ satisfying $\mathtt{DuIP}((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{y}_1, \mathbf{y}_2)) = 0$, it holds that:

$$
\begin{aligned}
&e(\ (x_{1,1}\odot C_{1,1})\cdots(x_{1,l_1}\odot C_{1,l_1})\cdot(x_{2,1}\odot C_{2,1})\cdots(x_{2,l_2}\odot C_{2,l_2}), K_1\ )\cdot e(C_0, K_0)^{-1} \\
=&e(\ [x_{1,1}\cdot\mathbf{s}^\top\mathbf{A}^\top(y_{1,1}\cdot\mathbf{U}_1 + \mathbf{W}_{1,1})]_1\ \cdots\ [x_{1,l_1}\cdot\mathbf{s}^\top\mathbf{A}^\top(y_{1,l_1}\cdot\mathbf{U}_1 + \mathbf{W}_{1,l_1})]_1 \\
&\cdot [x_{2,1}\cdot\mathbf{s}^\top\mathbf{A}^\top(y_{2,1}\cdot\mathbf{U}_2 + \mathbf{W}_{2,1})]_1\ \cdots\ [x_{2,l_2}\cdot\mathbf{s}^\top\mathbf{A}^\top(y_{2,l_2}\cdot\mathbf{U}_2 + \mathbf{W}_{2,l_2})]_1,\ [\mathbf{B}_1\mathbf{r}]_2\ )
\end{aligned}
$$

$\underline{\mathsf{Setup}(1^\lambda, l_1, l_2)}$:
$\mathbb{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$
$\mathbf{A} \leftarrow \mathbb{Z}_p^{(k+1)\times k}$, $\mathbf{B}_1 \leftarrow \mathbb{Z}_p^{(2k+1)\times k}$
$\mathbf{U}_1, \mathbf{U}_2, \mathbf{W}_{1,1}, \ldots, \mathbf{W}_{1,l_1}, \mathbf{W}_{2,1}, \ldots, \mathbf{W}_{2,l_2} \leftarrow \mathbb{Z}_p^{(k+1)\times(2k+1)}$, $\mathbf{k} \leftarrow \mathbb{Z}_p^{k+1}$
$pk := (\mathbb{G}, [\mathbf{A}^\top]_1, [\mathbf{A}^\top\mathbf{U}_1]_1, [\mathbf{A}^\top\mathbf{U}_2]_1, [\mathbf{A}^\top\mathbf{W}_{1,1}]_1, \ldots, [\mathbf{A}^\top\mathbf{W}_{1,l_1}]_1, [\mathbf{A}^\top\mathbf{W}_{2,1}]_1, \ldots, [\mathbf{A}^\top\mathbf{W}_{2,l_2}]_1, [\mathbf{A}^\top\mathbf{k}]_T)$
$msk := (\mathbf{k}, \mathbf{W}_{1,1}, \ldots, \mathbf{W}_{1,l_1}, \mathbf{W}_{2,1}, \ldots, \mathbf{W}_{2,l_2}, \mathbf{B}_1)$
Return (pk, msk)

$\underline{\mathsf{KGen}(msk, (\mathbf{x}_1, \mathbf{x}_2))}$:
Parse $\mathbf{x}_1 = (x_{1,1}, \ldots, x_{1,l_1}) \in \mathbb{Z}_p^{l_1}$, $\mathbf{x}_2 = (x_{2,1}, \ldots, x_{2,l_2}) \in \mathbb{Z}_p^{l_2}$
$\mathbf{r} \leftarrow \mathbb{Z}_p^k$
$K_0 := [\mathbf{k} + (x_{1,1} \cdot \mathbf{W}_{1,1} + \ldots + x_{1,l_1} \cdot \mathbf{W}_{1,l_1} + x_{2,1} \cdot \mathbf{W}_{2,1} + \ldots + x_{2,l_2} \cdot \mathbf{W}_{2,l_2})\mathbf{B}_1\mathbf{r}]_2$, $K_1 := [\mathbf{B}_1\mathbf{r}]_2$
Return $sk_{\mathbf{x}} := (K_0, K_1)$

$\underline{\mathsf{CheckKey}(pk, (\mathbf{x}_1, \mathbf{x}_2), sk_{\mathbf{x}})}$:
If $[\mathbf{0}]_T = e((x_{1,1} \odot [\mathbf{A}^\top\mathbf{W}_{1,1}]_1) \cdots (x_{1,l_1} \odot [\mathbf{A}^\top\mathbf{W}_{1,l_1}]_1) \cdot (x_{2,1} \odot [\mathbf{A}^\top\mathbf{W}_{2,1}]_1) \cdots (x_{2,l_2} \odot [\mathbf{A}^\top\mathbf{W}_{2,l_2}]_1), K_1)$
$\cdot e([\mathbf{A}^\top]_1, K_0)^{-1} \cdot [\mathbf{A}^\top\mathbf{k}]_T$: Return 1
Else Return 0

$\underline{\mathsf{Enc}(pk, (\mathbf{y}_1, \mathbf{y}_2), m \in \mathbb{G}_T)}$:
Parse $\mathbf{y}_1 = (y_{1,1}, \ldots, y_{1,l_1}) \in \mathbb{Z}_p^{l_1}$, $\mathbf{y}_2 = (y_{2,1}, \ldots, y_{2,l_2}) \in \mathbb{Z}_p^{l_2}$
$\mathbf{s} \leftarrow \mathbb{Z}_p^k$
$C_0 := [\mathbf{s}^\top\mathbf{A}^\top]_1$
For $i \in [l_1]$: $C_{1,i} := [\mathbf{s}^\top\mathbf{A}^\top(y_{1,i} \cdot \mathbf{U}_1 + \mathbf{W}_{1,i})]_1$
For $i \in [l_2]$: $C_{2,i} := [\mathbf{s}^\top\mathbf{A}^\top(y_{2,i} \cdot \mathbf{U}_2 + \mathbf{W}_{2,i})]_1$
$C := [\mathbf{s}^\top\mathbf{A}^\top\mathbf{k}]_T \cdot m$
Return $c_{\mathbf{y}} := (C_0, (C_{1,i})_{i\in[l_1]}, (C_{2,i})_{i\in[l_2]}, C)$

$\underline{\mathsf{Dec}(c_{\mathbf{y}}, sk_{\mathbf{x}})}$:
Parse $c_{\mathbf{y}} := (C_0, (C_{1,i})_{i\in[l_1]}, (C_{2,i})_{i\in[l_2]}, C)$, $sk_{\mathbf{x}} := (K_0, K_1)$
Return $m' := C \cdot e((x_{1,1} \odot C_{1,1}) \cdots (x_{1,l_1} \odot C_{1,l_1})(x_{2,1} \odot C_{2,1}) \cdots (x_{2,l_2} \odot C_{2,l_2}), K_1) \cdot e(C_0, K_0)^{-1}$

Fig. 9: The algorithms of public-key dual IPE scheme pkDIPE

$$
\begin{aligned}
&\cdot e(\ [\mathbf{s}^\top\mathbf{A}^\top]_1,\ [\mathbf{k} + (x_{1,1} \cdot \mathbf{W}_{1,1} + \cdots + x_{1,l_1} \cdot \mathbf{W}_{1,l_1} + x_{2,1} \cdot \mathbf{W}_{2,1} + \cdots + x_{2,l_2} \cdot \mathbf{W}_{2,l_2})\mathbf{B}_1\mathbf{r}]_2\ )^{-1} \\
=&[\langle \mathbf{x}_1, \mathbf{y}_1\rangle \cdot \mathbf{s}^\top\mathbf{A}^\top\mathbf{U}_1\mathbf{B}_1\mathbf{r} + \langle \mathbf{x}_2, \mathbf{y}_2\rangle \cdot \mathbf{s}^\top\mathbf{A}^\top\mathbf{U}_2\mathbf{B}_1\mathbf{r}]_T \cdot [\mathbf{s}^\top\mathbf{A}^\top\mathbf{k}]_T^{-1} \\
&\cdot [\mathbf{s}^\top\mathbf{A}^\top(x_{1,1} \cdot \mathbf{W}_{1,1} + \cdots + x_{1,l_1} \cdot \mathbf{W}_{1,l_1} + x_{2,1} \cdot \mathbf{W}_{2,1} + \cdots + x_{2,l_2} \cdot \mathbf{W}_{2,l_2})\mathbf{B}_1\mathbf{r}]_T \\
&\cdot [\mathbf{s}^\top\mathbf{A}^\top(x_{1,1} \cdot \mathbf{W}_{1,1} + \cdots + x_{1,l_1} \cdot \mathbf{W}_{1,l_1} + x_{2,1} \cdot \mathbf{W}_{2,1} + \cdots + x_{2,l_2} \cdot \mathbf{W}_{2,l_2})\mathbf{B}_1\mathbf{r}]_T^{-1} \\
=&[\mathbf{s}^\top\mathbf{A}^\top\mathbf{k}]_T^{-1}
\end{aligned}
$$

The above equation holds because of $\langle \mathbf{x}_1, \mathbf{y}_1\rangle = 0$ and $\langle \mathbf{x}_2, \mathbf{y}_2\rangle = 0$.

Thus, the decryption algorithm Dec outputs a correct message, $m' = [\mathbf{s}^\top\mathbf{A}^\top\mathbf{k}]_T \cdot m \cdot [\mathbf{s}^\top\mathbf{A}^\top\mathbf{k}]_T^{-1} = m$.

In addition, it also holds that

$$
\begin{aligned}
&e((x_{1,1} \odot [\mathbf{A}^\top\mathbf{W}_{1,1}]_1) \cdots (x_{1,l_1} \odot [\mathbf{A}^\top\mathbf{W}_{1,l_1}]_1) \\
&\quad \cdot (x_{2,1} \odot [\mathbf{A}^\top\mathbf{W}_{2,1}]_1) \cdots (x_{2,l_2} \odot [\mathbf{A}^\top\mathbf{W}_{2,l_2}]_1), K_1) \cdot e([\mathbf{A}^\top]_1, K_0)^{-1} \cdot [\mathbf{A}^\top\mathbf{k}]_T \\
=&e([x_{1,1} \cdot \mathbf{A}^\top\mathbf{W}_{1,1}]_1 \cdots [x_{1,l_1} \cdot \mathbf{A}^\top\mathbf{W}_{1,l_1}]_1 \cdot [x_{2,1} \cdot \mathbf{A}^\top\mathbf{W}_{2,1}]_1 \cdots [x_{2,l_2} \cdot \mathbf{A}^\top\mathbf{W}_{2,l_2}]_1, [\mathbf{B}_1\mathbf{r}]_2) \\
&\quad \cdot e([\mathbf{A}^\top]_1, [\mathbf{k} + (x_{1,1} \cdot \mathbf{W}_{1,1} + \cdots + x_{1,l_1} \cdot \mathbf{W}_{1,l_1} + x_{2,1} \cdot \mathbf{W}_{2,1} + \cdots + x_{2,l_2} \cdot \mathbf{W}_{2,l_2})\mathbf{B}_1\mathbf{r}]_2)^{-1}
\end{aligned}
$$

$$
\begin{aligned}
&\cdot [\mathbf{A}^\top \mathbf{k}]_T \\
=&e([x_{1,1} \cdot \mathbf{A}^\top \mathbf{W}_{1,1}]_1 \cdots [x_{1,l_1} \cdot \mathbf{A}^\top \mathbf{W}_{1,l_1}]_1 \cdot [x_{2,1} \cdot \mathbf{A}^\top \mathbf{W}_{2,1}]_1 \cdots [x_{2,l_2} \cdot \mathbf{A}^\top \mathbf{W}_{2,l_2}]_1, [\mathbf{B}_1 \mathbf{r}]_2) \\
&\cdot e([\mathbf{A}^\top]_1, [(x_{1,1} \cdot \mathbf{W}_{1,1} + \cdots + x_{1,l_1} \cdot \mathbf{W}_{1,l_1} + x_{2,1} \cdot \mathbf{W}_{2,1} + \cdots + x_{2,l_2} \cdot \mathbf{W}_{2,l_2}) \mathbf{B}_1 \mathbf{r}]_2)^{-1} \\
&\cdot e([\mathbf{A}^\top]_1, [\mathbf{k}]_2)^{-1} \cdot [\mathbf{A}^\top \mathbf{k}]_T \\
=&[(x_{1,1} \cdot \mathbf{A}^\top \mathbf{W}_{1,1} + \cdots + x_{1,l_1} \cdot \mathbf{A}^\top \mathbf{W}_{1,l_1} + x_{2,1} \cdot \mathbf{A}^\top \mathbf{W}_{2,1} + \cdots + x_{2,l_2} \cdot \mathbf{A}^\top \mathbf{W}_{2,l_2}) \mathbf{B}_1 \mathbf{r}]_T \\
&\cdot [(x_{1,1} \cdot \mathbf{A}^\top \mathbf{W}_{1,1} + \cdots + x_{1,l_1} \cdot \mathbf{A}^\top \mathbf{W}_{1,l_1} + x_{2,1} \cdot \mathbf{A}^\top \mathbf{W}_{2,1} + \cdots + x_{2,l_2} \cdot \mathbf{A}^\top \mathbf{W}_{2,l_2}) \mathbf{B}_1 \mathbf{r}]_T^{-1} \\
&\cdot [\mathbf{A}^\top \mathbf{k}]_T^{-1} \cdot [\mathbf{A}^\top \mathbf{k}]_T \\
=&[\mathbf{0}]_T.
\end{aligned}
$$

Thus, the checking algorithm $\mathsf{CheckKey}$ outputs 1.

In all, the dual IPE constructed in Fig. 9 is correct.

Security. We now state the security theorem of our proposed public-key dual IPE scheme. Due to space limitations, the proof is given in Appendix J.

Theorem 6. *Under the k-LIN assumption and the adaptive secuity of the private-key dual IPE scheme described in Fig. 8, the public-key dual IPE scheme described in Fig. 9 is adaptively secure.*

5.4 A SME supporting Sigma protocol

In Sec. 5.2, we show how to construct a SME from a dual IPE. Thus, leveraging the concrete dual IPE in Sec. 5.3, we can obtain a concrete SME. Note that the dual IPE DIPE constructed in Sec. 5.3 is based on the k-LIN assumption (here we pick $k = 1$, i.e., based on the SXDH assumption [12]). In the following, we show that the concrete SME supports Sigma protocols.

We firstly show the main idea of how to transfer the relation $\mathcal{R}_c$ when the underlying SME scheme is constructed from the dual IPE in Sec. 5.3. After that, we give the details of the Sigma protocol.

Main idea. Recall that the relation of the property of supporting Sigma protocols is as follows:

$$\mathcal{R}_c = \{((\tau_{\mathsf{p}}, c, pk, \Phi), (w, r_{\mathrm{enc}})) : c = \mathsf{SME.Enc}(pk, \Phi, \tau_{\mathsf{p}}, w; r_{\mathrm{enc}})\}.$$

As shown in Fig. 7, we can know that $\mathsf{SME.Enc}$ mainly invokes the encryption algorithm of dual IPE (i.e., $\mathsf{DIPE.Enc}$). Note that when $w \notin \Phi$, $\mathsf{SME.Enc}$ would output $\perp$. Thus, directly proving that c is the output by the encryption of DIPE is not enough, since it is also required to guarantee $w \in \Phi$.

A simple idea is to adopt the "OR" technique, such that we can prove that there exists a w in Φ and the ciphertext c for a default message $\mathsf{m}_{\mathtt{dum}}$ is generated with a predicate corresponding to w.

Then, what remains is to prove that the ciphertext is well-formed when given a $w \in \Phi$. As introduced in the aforementioned section, the witness w is transferred to $I(w)$. With respect to the algorithm $\mathsf{EncodeW}$, we can know that $I(w)$

only contains a one and other positions are labelled by zeros. Therefore, when adopting DIPE to construct SME, roughly $\mathcal{R}_c$ can be transferred as follows:

$$\mathcal{R}_c^{\mathsf{DIPE}} = \{((\tau_{\mathtt{p}}, c, pk, \Phi), r_{\mathrm{enc}}) : \vee_{w\in\Phi}(\ (\ c = \mathsf{DIPE.Enc}(pk, (I(w), \tau_{\mathtt{p}}), \mathsf{m}_{\mathtt{dum}}; r_{\mathrm{enc}}), \text{ where } I(w) \text{ has a one and other positions are zeros }\ )\}.$$

Details of the Sigma protocol. Plugging with the concrete algorithms of DIPE, we can transfer the relation $\mathcal{R}_c^{\mathsf{DIPE}}$ into the following relation $\mathcal{R}'$.

$$\begin{aligned}
\mathcal{R}' = \{((\tau_{\mathtt{p}}, c, pk, \mathsf{m}_{\mathtt{dum}}, \Phi), \mathbf{s}) : & \quad /\!/\ |\Phi| = l_1 \\
(\vee_{i\in[l_1]}\ (\ \ & (C_0 = [\mathbf{s}^\top \mathbf{A}^\top]_1) \wedge (C = [\mathbf{s}^\top \mathbf{A}^\top \mathbf{k}]_T \cdot \mathsf{m}_{\mathtt{dum}}) \\
& \wedge (C_{1,1} = [\mathbf{s}^\top \mathbf{A}^\top \mathbf{W}_{1,1}]_1)\ \wedge \ldots & /\!/\ I(w)_1 = 0 \ldots \\
& \wedge (C_{1,i-1} = [\mathbf{s}^\top \mathbf{A}^\top \mathbf{W}_{1,i-1}]_1) & /\!/\ I(w)_{i-1} = 0 \\
& \wedge (C_{1,i} = [\mathbf{s}^\top \mathbf{A}^\top (\mathbf{U}_1 + \mathbf{W}_{1,i})]_1) & /\!/\ I(w)_i = 1 \\
& \wedge (C_{1,i+1} = [\mathbf{s}^\top \mathbf{A}^\top \mathbf{W}_{1,i+1}]_1)\ \wedge\ \ldots & /\!/\ I(w)_{i+1} = 0 \ldots \\
& \wedge (C_{1,l_1} = [\mathbf{s}^\top \mathbf{A}^\top \mathbf{W}_{1,l_1}]_1) & /\!/\ I(w)_{l_1} = 0 \\
& \wedge\ (\wedge_{\iota\in[l_2]}(C_{2,\iota} = [\mathbf{s}^\top \mathbf{A}^\top ((\tau_{\mathtt{p}})_\iota \mathbf{U}_2 + \mathbf{W}_{2,\iota})]_1))\ )\ )\} & /\!/\ \tau_{\mathtt{p}}
\end{aligned}$$

Then, for each clause, we show that we can construct a Sigma protocol. We denote the j^{th} clause in the relation $\mathcal{R}'$ as $\mathcal{R}'_j$, as shown in the following.

$$\begin{aligned}
\mathcal{R}'_j = \{((\tau_{\mathtt{p}}, c, pk, \mathsf{m}_{\mathtt{dum}}, j), \mathbf{s}) : & \\
& (C_0 = [\mathbf{s}^\top \mathbf{A}^\top]_1) \wedge (C = [\mathbf{s}^\top \mathbf{A}^\top \mathbf{k}]_T \cdot \mathsf{m}_{\mathtt{dum}}) \\
& \wedge (C_{1,1} = [\mathbf{s}^\top \mathbf{A}^\top \mathbf{W}_{1,1}]_1)\ \wedge \ldots & /\!/\ I(w)_1 = 0 \ldots \\
& \wedge (C_{1,j-1} = [\mathbf{s}^\top \mathbf{A}^\top \mathbf{W}_{1,j-1}]_1) & /\!/\ I(w)_{j-1} = 0 \\
& \wedge (C_{1,j} = [\mathbf{s}^\top \mathbf{A}^\top (\mathbf{U}_1 + \mathbf{W}_{1,j})]_1) & /\!/\ I(w)_j = 1 \\
& \wedge (C_{1,j+1} = [\mathbf{s}^\top \mathbf{A}^\top \mathbf{W}_{1,j+1}]_1)\ \wedge\ \ldots & /\!/\ I(w)_{j+1} = 0 \ldots \\
& \wedge (C_{1,l_1} = [\mathbf{s}^\top \mathbf{A}^\top \mathbf{W}_{1,l_1}]_1) & /\!/\ I(w)_{l_1} = 0 \\
& \wedge\ (\wedge_{\iota\in[l_2]}(C_{2,\iota} = [\mathbf{s}^\top \mathbf{A}^\top ((\tau_{\mathtt{p}})_\iota \mathbf{U}_2 + \mathbf{W}_{2,\iota})]_1))\ \} & /\!/\ \tau_{\mathtt{p}}
\end{aligned}$$

We present our Sigma protocol $\Sigma^{\mathcal{R}'}_{\mathrm{clause}} = (\mathcal{P}, \mathcal{V})$ in Fig. 10 for the relation $\mathcal{R}'_j$, which essentially is an extension of the Chaum-Pedersen protocol [8].

Due to space limitations, the security analysis of $\Sigma^{\mathcal{R}'}_{\mathrm{clause}}$ is placed in Appendix K. Recall that we set $k = 1$ for the underlying dual IPE (i.e., based on the SXDH assumption), the communication overhead is listed as follows: 1) the commitment a output by $\mathcal{P}_1$ contains $3(l_1+l_2)+2$ elements in $\mathbb{G}_1$ and 1 element in $\mathbb{G}_T$; 2) the challenge c selected by $\mathcal{V}_2$ contains 1 element in $\mathbb{Z}_p$; 3) the response z generated by $\mathcal{P}_2$ contains 1 element in $\mathbb{Z}_p$.

Therefore, the scheme SME constructed above supports Sigma protocols.

(1) $\underline{\mathcal{P}_1(x = (\tau_{\mathsf{p}}, c, pk, \mathsf{m}_{\mathtt{dum}}, j), w = \mathbf{s})}$:
$\mathbf{t} \leftarrow \mathbb{Z}_p$, $\mathsf{a}_{C_0} := [\mathbf{t}^\top \mathbf{A}^\top]_1$, $\mathsf{a}_{C_{1,j}} := [\mathbf{t}^\top \mathbf{A}^\top (\mathbf{U}_1 + \mathbf{W}_{1,j})]_1$
For each $i \in [l_1] \backslash \{j\}$: $\mathsf{a}_{C_{1,i}} := [\mathbf{t}^\top \mathbf{A}^\top \mathbf{W}_{1,i})]_1$
For each $\iota \in [l_2]$: $\mathsf{a}_{C_{2,\iota}} := [\mathbf{t}^\top \mathbf{A}^\top ((\tau_{\mathsf{p}})_\iota \cdot \mathbf{U}_2 + \mathbf{W}_{2,\iota}))]_1$
$\mathsf{a}_C := [\mathbf{t}^\top \mathbf{A}^\top \mathbf{k}]_T$, send $\mathsf{a} := (\mathsf{a}_{C_0}, (\mathsf{a}_{C_{1,i}})_{i \in [l_1]}, (\mathsf{a}_{C_{2,\iota}})_{\iota \in [l_2]}, \mathsf{a}_C)$ to $\mathcal{V}$
(2) $\underline{\mathcal{V}_1(\mathsf{a})}$: $\mathsf{c} \leftarrow \mathbb{Z}_p$, send c to $\mathcal{P}$
(3) $\underline{\mathcal{P}_2(\mathsf{a}, \mathsf{c}, x, w)}$: $\mathbf{z_t} := \mathbf{t} - \mathsf{c} \cdot \mathbf{s}$, send $\mathsf{z} := \mathbf{z_t}$ to $\mathcal{V}$

$\underline{\mathcal{V}_2(x, \mathsf{a}, \mathsf{c}, \mathsf{z})}$:
Parse $x = (\tau_{\mathsf{p}}, c, pk, \mathsf{m}_{\mathtt{dum}}, j)$, $\mathsf{a} = (\mathsf{a}_{C_0}, (\mathsf{a}_{C_{1,i}})_{i \in [l_1]}, (\mathsf{a}_{C_{2,\iota}})_{\iota \in [l_2]}, \mathsf{a}_C)$
$\mathbf{z_t} := \mathsf{z}$, $\mathsf{a}'_{C_0} := [\mathbf{z_t}^\top \mathbf{A}^\top]_1 (\mathsf{c} \odot C_0)$, $\mathsf{a}'_{C_{1,j}} := [\mathbf{z_t}^\top \mathbf{A}^\top (\mathbf{U}_1 + \mathbf{W}_{1,j})]_1 (\mathsf{c} \odot C_{1,j})$
For each $i \in [l_1] \backslash \{j\}$: $\mathsf{a}'_{C_{1,i}} := [\mathbf{z_t}^\top \mathbf{A}^\top \mathbf{W}_{1,i}]_1 (\mathsf{c} \odot C_{1,i})$
For each $\iota \in [l_2]$: $\mathsf{a}'_{C_{2,\iota}} := [\mathbf{z_t}^\top \mathbf{A}^\top ((\tau_{\mathsf{p}})_\iota \cdot \mathbf{U}_2 + \mathbf{W}_{2,\iota})]_1 (\mathsf{c} \odot C_{2,\iota})$
$\mathsf{a}'_C := [\mathbf{z_t}^\top \mathbf{A}^\top \mathbf{k}]_T \cdot (\mathsf{c} \odot (C / \mathsf{m}_{\mathtt{dum}}))$
If $(\mathsf{a}'_{C_0} = \mathsf{a}_{C_0}) \wedge (\mathsf{a}'_{C_{1,i}} = \mathsf{a}_{C_{1,i}})_{i \in [l_1]} \wedge (\mathsf{a}'_{C_{2,\iota}} = \mathsf{a}_{C_{2,\iota}})_{\iota \in [l_2]} \wedge (\mathsf{a}'_C = \mathsf{a}_C)$: Return 1
Else Return 0

Fig. 10: Algorithms of $\Sigma^{\mathcal{R}'}_{\text{clause}} = (\mathcal{P}, \mathcal{V})$

5.5 A more efficient fSMP

Now we discuss a concrete construction for fSMP. More exactly, we adopt Pedersen commitment [27] (for simplicity, the commitment scheme runs over the group $\mathbb{G}_{\mathsf{com}}$ with prime order p, and g and h are two generators of group $\mathbb{G}_{\mathsf{com}}$ where $\log_g h$ is unknown) and the SME scheme introduced in Sec. 5.2 (constructed with a concrete dual IPE introduced in Sec. 5.3).

Firstly, we show that we can construct a Sigma protocol for the internal relation $\widetilde{\mathcal{R}_{\text{sm}}}$ (see in the following). It implies a NIZK for $\widetilde{\mathcal{R}_{\text{sm}}}$ by applying the Fiat-Shamir transform. The size of NIZK proof would be $O(l_1 \cdot \mathsf{poly}(\lambda))$, where l_1 is the size of the set Φ and λ is the security parameter. After that, we discuss how to improve the size of proof for relation $\widetilde{\mathcal{R}_{\text{sm}}}$. Applying self-stacking technique [18], we will show that the size of proof can be logarithmic, i.e., $O((\log l_1) \cdot \mathsf{poly}(\lambda)) = O(\log l_1)$.

A sigma protocol for $\widetilde{\mathcal{R}_{\text{sm}}}$. Recall that the relation $\widetilde{\mathcal{R}_{\text{sm}}}$ is as follows:

$$\widetilde{\mathcal{R}_{\text{sm}}} = \{((\tau_{\mathsf{p}}, \mathsf{com}, c, pk, \Phi), (w, r_{\text{com}}, r_{\text{enc}})) : \mathsf{com} = \mathsf{Commit.Com}(pp, w; r_{\text{com}}) \wedge c = \mathsf{SME.Enc}(pk, \Phi, (w, \tau_{\mathsf{p}}); r_{\text{enc}})\}.$$

Following the transformation discussed in Sec. 5.4, the relation $\widetilde{\mathcal{R}_{\text{sm}}}$ can be transferred into the following relation $\mathcal{R}''$.

$\mathcal{R}'' = \{((g, h, \tau_{\mathsf{p}}, \mathsf{com}, c, pk, \mathsf{m}_{\mathtt{dum}}, \Phi), (r_{\text{com}}, \mathbf{s})) :$
$(\vee_{i \in [l_1]} (\ (\mathsf{com} = g^{\Phi_i} h^{r_{\text{com}}})$ // Φ_i is the i^{th} element in Φ

$$
\begin{aligned}
&\wedge\ (C_0 = [\mathbf{s}^\top \mathbf{A}^\top]_1) \wedge (C = [\mathbf{s}^\top \mathbf{A}^\top \mathbf{k}]_T \cdot \mathsf{m}_{\mathtt{dum}}) && \\
&\wedge (C_{1,1} = [\mathbf{s}^\top \mathbf{A}^\top \mathbf{W}_{1,1}]_1)\ \wedge \ldots && /\!/\ I(w)_1 = 0 \ldots \\
&\wedge (C_{1,i-1} = [\mathbf{s}^\top \mathbf{A}^\top \mathbf{W}_{1,i-1}]_1) && /\!/\ I(w)_{i-1} = 0 \\
&\wedge (C_{1,i} = [\mathbf{s}^\top \mathbf{A}^\top (\mathbf{U}_1 + \mathbf{W}_{1,i})]_1) && /\!/\ I(w)_i = 1 \\
&\wedge (C_{1,i+1} = [\mathbf{s}^\top \mathbf{A}^\top \mathbf{W}_{1,i+1}]_1)\ \wedge\ \ldots && /\!/\ I(w)_{i+1} = 0 \ldots \\
&\wedge (C_{1,l_1} = [\mathbf{s}^\top \mathbf{A}^\top \mathbf{W}_{1,l_1}]_1) && /\!/\ I(w)_{l_1} = 0 \\
&\wedge\ (\wedge_{\iota \in [l_2]} (C_{2,\iota} = [\mathbf{s}^\top \mathbf{A}^\top ((\tau_{\mathtt{p}})_\iota \mathbf{U}_2 + \mathbf{W}_{2,\iota})]_1))\ )\ )\} && /\!/\ \tau_{\mathtt{p}}
\end{aligned}
$$

It is clear that we can prove $\mathcal{R}''$ by the composite Sigma protocol. More exactly, we can construct a Sigma protocol for $\mathcal{R}''$ through the following method:

1. Firstly, for each clause $\mathcal{R}''_j$ ($j \in [l_1]$) in $\mathcal{R}''$, we can construct a Sigma protocol $\Sigma^{\mathcal{R}''}_{\text{clause}}$ by adopting the "AND" technique to composite Schnorr's Sigma protocol for the discrete logarithm of the commitment (i.e., knowing the discrete logarithm of $(\mathsf{com}/g^{\Phi_i})$ with base h is r_{com}) and the Sigma protocol $\Sigma^{\mathcal{R}'}_{\text{clause}}$ for the SME.
2. Secondly, we can obtain a composite Sigma protocol for relation $\mathcal{R}''$ by adopting the "OR" technique to composite the Sigma protocols for all clause.

We present a Sigma protocol $\Sigma^{\mathcal{R}''}_{\text{clause}}$ for a clause $\mathcal{R}''_j$ (as defined below) in Fig. 11 .

$$
\begin{aligned}
\mathcal{R}''_j = \{((g, h, \tau_{\mathtt{p}}, \mathsf{com}, c, pk, \mathsf{m}_{\mathtt{dum}}, \Phi_j, j), (r_{\text{com}}, \mathbf{s})) :& \\
(\mathsf{com} = g^{\Phi_j} h^{r_{\text{com}}}) && /\!/\ \Phi_j \text{ is the } j^{th} \text{ element in } \Phi \\
\wedge (C_0 = [\mathbf{s}^\top \mathbf{A}^\top]_1) \wedge (C = [\mathbf{s}^\top \mathbf{A}^\top \mathbf{k}]_T \cdot \mathsf{m}_{\mathtt{dum}}) & \\
\wedge (C_{1,1} = [\mathbf{s}^\top \mathbf{A}^\top \mathbf{W}_{1,1}]_1)\ \wedge \ldots && /\!/\ I(w)_1 = 0 \ldots \\
\wedge (C_{1,j-1} = [\mathbf{s}^\top \mathbf{A}^\top \mathbf{W}_{1,j-1}]_1) && /\!/\ I(w)_{j-1} = 0 \\
\wedge (C_{1,j} = [\mathbf{s}^\top \mathbf{A}^\top (\mathbf{U}_1 + \mathbf{W}_{1,j})]_1) && /\!/\ I(w)_j = 1 \\
\wedge (C_{1,j+1} = [\mathbf{s}^\top \mathbf{A}^\top \mathbf{W}_{1,j+1}]_1)\ \wedge\ \ldots && /\!/\ I(w)_{j+1} = 0 \ldots \\
\wedge (C_{1,l_1} = [\mathbf{s}^\top \mathbf{A}^\top \mathbf{W}_{1,l_1}]_1) && /\!/\ I(w)_{l_1} = 0 \\
\wedge\ (\wedge_{\iota \in [l_2]} (C_{2,\iota} = [\mathbf{s}^\top \mathbf{A}^\top ((\tau_{\mathtt{p}})_\iota \mathbf{U}_2 + \mathbf{W}_{2,\iota})]_1))\ \} && /\!/\ \tau_{\mathtt{p}}
\end{aligned}
$$

Since the Sigma protocol $\Sigma^{\mathcal{R}''}_{\text{clause}}$ for $\mathcal{R}''_j$ is obtained by compositing Schnorr's Sigma protocol and $\Sigma^{\mathcal{R}'}_{\text{clause}}$ for relation $\mathcal{R}'_j$ (as shown in Sec. 5.4) using an "AND" combination, the security of $\Sigma^{\mathcal{R}''}_{\text{clause}}$ is well guaranteed. Here we omit the discussions on its security.

Recall that we set $k = 1$ for the underlying dual IPE (i.e., based on the SXDH assumption), so the communication overhead of $\Sigma^{\mathcal{R}''}_{\text{clause}}$ would be 1 element in $\mathbb{G}_{\mathsf{com}}$, $3(l_1 + l_2) + 2$ elements in $\mathbb{G}_1$, 1 element in $\mathbb{G}_T$ and 3 elements in $\mathbb{Z}_p$. Thus, it can be inferred that using the conventional composition method, the total communication overhead would be $O(l_1(l_1 + l_2))$ (since there are l_1 clauses). By

(1) $\mathcal{P}_1(x = (g, h, \tau_{\mathsf{p}}, \mathsf{com}, c, pk, \mathsf{m}_{\mathtt{dum}}, \Phi_j, j), w = (r_{\mathrm{com}}, \mathbf{s}))$:
$t_{\mathsf{com}} \leftarrow \mathbb{Z}_p$, $\mathsf{a}_{\mathsf{com}} := h^{t_{\mathsf{com}}}$
$\mathbf{t} \leftarrow \mathbb{Z}_p$, $\mathsf{a}_{C_0} := [\mathbf{t}^\top \mathbf{A}^\top]_1$, $\mathsf{a}_{C_{1,j}} := [\mathbf{t}^\top \mathbf{A}^\top (\mathbf{U}_1 + \mathbf{W}_{1,j})]_1$
For each $i \in [l_1] \backslash \{j\}$: $\mathsf{a}_{C_{1,i}} := [\mathbf{t}^\top \mathbf{A}^\top \mathbf{W}_{1,i})]_1$
For each $\iota \in [l_2]$: $\mathsf{a}_{C_{2,\iota}} := [\mathbf{t}^\top \mathbf{A}^\top ((\tau_{\mathsf{p}})_\iota \cdot \mathbf{U}_2 + \mathbf{W}_{2,\iota}))]_1$
$\mathsf{a}_C := [\mathbf{t}^\top \mathbf{A}^\top \mathbf{k}]_T$, send $\mathsf{a} := (\mathsf{a}_{\mathsf{com}}, \mathsf{a}_{C_0}, (\mathsf{a}_{C_{1,i}})_{i \in [l_1]}, (\mathsf{a}_{C_{2,\iota}})_{\iota \in [l_2]}, \mathsf{a}_C)$ to $\mathcal{V}$
(2) $\mathcal{V}_1(\mathsf{a})$: $\mathsf{c} \leftarrow \mathbb{Z}_p$, send c to $\mathcal{P}$
(3) $\mathcal{P}_2(\mathsf{a}, \mathsf{c}, x, w)$: $\mathsf{z}_{\mathsf{com}} := t_{\mathsf{com}} - \mathsf{c} \cdot r_{\mathrm{com}}$, $\mathbf{z_t} := \mathbf{t} - \mathsf{c} \cdot \mathbf{s}$, send $\mathsf{z} := (\mathsf{z}_{\mathsf{com}}, \mathbf{z_t})$ to $\mathcal{V}$

$\mathcal{V}_2(x, \mathsf{a}, \mathsf{c}, \mathsf{z})$:
Parse $x = (g, h, \tau_{\mathsf{p}}, \mathsf{com}, \mathsf{c}, pk, \mathsf{m}_{\mathtt{dum}}, \Phi_j, j)$
Parse $\mathsf{a} = (\mathsf{a}_{\mathsf{com}}, \mathsf{a}_{C_0}, (\mathsf{a}_{C_{1,i}})_{i \in [l_1]}, (\mathsf{a}_{C_{2,\iota}})_{\iota \in [l_2]}, \mathsf{a}_C)$, $\mathsf{z} = (\mathsf{z}_{\mathsf{com}}, \mathbf{z_t})$
$\mathsf{a}'_{\mathsf{com}} := h^{\mathsf{z}_{\mathsf{com}}} (\mathsf{com}/g^{\Phi_j})^{\mathsf{c}}$, $\mathsf{a}'_{C_0} := [\mathbf{z_t}^\top \mathbf{A}^\top]_1 (\mathsf{c} \odot C_0)$
$\mathsf{a}'_{C_{1,j}} := [\mathbf{z_t}^\top \mathbf{A}^\top (\mathbf{U}_1 + \mathbf{W}_{1,j})]_1 (\mathsf{c} \odot C_{1,j})$
For each $i \in [l_1] \backslash \{j\}$: $\mathsf{a}'_{C_{1,i}} := [\mathbf{z_t}^\top \mathbf{A}^\top \mathbf{W}_{1,i}]_1 (\mathsf{c} \odot C_{1,i})$
For each $\iota \in [l_2]$: $\mathsf{a}'_{C_{2,\iota}} := [\mathbf{z_t}^\top \mathbf{A}^\top ((\tau_{\mathsf{p}})_\iota \cdot \mathbf{U}_2 + \mathbf{W}_{2,\iota})]_1 (\mathsf{c} \odot C_{2,\iota})$
$\mathsf{a}'_C := [\mathbf{z_t}^\top \mathbf{A}^\top \mathbf{k}]_T \cdot (\mathsf{c} \odot (C/\mathsf{m}_{\mathtt{dum}}))$
If $(\mathsf{a}'_{\mathsf{com}} = \mathsf{a}_{\mathsf{com}}) \wedge (\mathsf{a}'_{C_0} = \mathsf{a}_{C_0}) \wedge (\mathsf{a}'_{C_{1,i}} = \mathsf{a}_{C_{1,i}})_{i \in [l_1]} \wedge (\mathsf{a}'_{C_{2,\iota}} = \mathsf{a}_{C_{2,\iota}})_{\iota \in [l_2]}$
$\wedge (\mathsf{a}'_C = \mathsf{a}_C)$: Return 1
Else Return 0

Fig. 11: Algorithms of $\Sigma^{\mathcal{R}''}_{\mathrm{clause}} = (\mathcal{P}, \mathcal{V})$

applying the Fiat-Shamir transform to the composited Sigma protocol[7], we can obtain a NIZK for $\widetilde{\mathcal{R}_{\mathrm{sm}}}$ and the proof size would be $3l_1$ elements in $\mathbb{Z}_p$, where $2l_1$ elements are the responses (i.e., all z's for all clauses) and l_1 elements are the challenges for all clauses.

Proofs with shorter size. In [18], Goel et al. present a general framework for composing *stackable Sigma protocols* for disjunctions in which communication depends on the size of the largest clause. Notably, they also demonstrate the stackability of several classic Sigma protocols, including Schnorr's Sigma protocol [30] and the Chaum-Pedersen protocol [8].

We will show that the Sigma protocol $\Sigma^{\mathcal{R}''}_{\mathrm{clause}}$ is stackable. Then, according to Theorem 8 (in Appendix A.3), we can achieve a compiled Sigma protocol by utilizing the self-stacking technique [18] and we can know that its communication complexity is linearly proportional to the largest communication cost among these clauses. Note that in $\mathcal{R}''$, we can run $\Sigma^{\mathcal{R}''}_{\mathrm{clause}}$ for every clause, so every clause has the same communication cost. Thus, the communication complexity

[7] Note that the Sigma protocol $\Sigma^{\mathcal{R}''}_{\mathrm{clause}}$ supports that the verifier can recover the commitment via the responses and the challenge, so we can save the proof size by only sending the responds to the verifier as the proof.

of the compiled Sigma protocol is linearly proportional to the communication overhead of $\Sigma^{\mathcal{R}''}_{\text{clause}}$.

We have the following theorem. Due to space limitations, the definition of stackable Sigma protocol is recalled in Appendix A.3, and the proof for Theorem 7 is provided in Appendix L.

Theorem 7. *The Sigma protocol $\Sigma^{\mathcal{R}''}_{\text{clause}}$ in Fig. 11 is a stackable Sigma protocol.*

Communication complexity. Here, we follow the analysis in [18]. Let $\mathsf{CC}(\Sigma)$ be the communication complexity of $\Sigma^{\mathcal{R}''}_{\text{clause}}$ for the relation $\mathcal{R}''_j$ and Σ_{l_1} denote the compiled Sigma protocol for $\mathcal{R}''$. Let $|\mathsf{Size}(\mathsf{VCommit})|$ be the size of the 1-out-of-l binding vector commitment scheme (please refer its definition to [18]), which is independent of $\mathsf{CC}(\Sigma)$ and only depends on the security parameter λ. We have $\mathsf{CC}(\Sigma_{l_1}) = \mathsf{CC}(\Sigma) + 2(\log l_1)(|\mathsf{Size}(\mathsf{VCommit})|) = O(\mathsf{CC}(\Sigma) + (\log l_1) \cdot \mathsf{poly}(\lambda))$. Thus, when applying the Fiat-Shamir transform, the proof size would be $O((\log l_1) \cdot \mathsf{poly}(\lambda)) = O(\log l_1)$.

Acknowledgements

We would like to express our sincere appreciation to Junqing Gong for his valuable suggestions on the inner-product encryption! We also want to express our sincere appreciation to the anonymous reviewers for their valuable comments and suggestions!

References

1. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: STOC 1988. pp. 103–112. ACM (1988)
2. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: TCC 2011. pp. 253–273. Springer (2011)
3. Boneh, D., Shoup, V.: A graduate course in applied cryptography. Draft 0.6 (2023)
4. Bunz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short Proofs for Confidential Transactions and More. In: S&P 2018. pp. 315–334. IEEE (2018)
5. Camenisch, J., Chaabouni, R., Shelat, A.: Efficient protocols for set membership and range proofs. In: ASIACRYPT 2008. pp. 234–252. Springer (2008)
6. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact e-cash. In: EUROCRYPT 2005. pp. 302–321. Springer (2005)
7. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: EUROCRYPT 2001. pp. 93–118. Springer (2001)
8. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: CRYPTO 1992. pp. 89–105. Springer (1992)
9. Chaum, D., Van Heyst, E.: Group signatures. In: EUROCRYPT 1991. pp. 257–265. Springer (1991)

10. 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)
11. Chen, J., Gong, J., Kowalczyk, L., Wee, H.: Unbounded abe via bilinear entropy expansion, revisited. In: EUROCRYPT 2018. pp. 503–534. Springer (2018)
12. Chen, J., Gong, J., Wee, H.: Improved inner-product encryption with adaptive security and full attribute-hiding. In: ASIACRYPT 2018. pp. 673–702. Springer (2018)
13. Couteau, G., Klooß, M., Lin, H., Reichle, M.: Efficient range proofs with transparent setup from bounded integer commitments. In: EUROCRYPT 2021. pp. 247–277. Springer (2021)
14. Cramer, R., Damgård, I.: Zero-knowledge proofs for finite field arithmetic, or: Can zero-knowledge be for free? In: CRYPTO 1998. pp. 424–441. Springer (1998)
15. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for diffie–hellman assumptions. J. Cryptology **30**, 242–288 (2017)
16. Feige, U., Lapidot, D., Shamir, A.: Multiple noninteractive zero knowledge proofs under general assumptions. SIAM J. Comput. **29**(1), 1–28 (1999)
17. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: CRYPTO 1986. pp. 186–194. Springer (1986)
18. Goel, A., Green, M., Hall-Andersen, M., Kaptchuk, G.: Stacking Sigmas: A Framework to Compose Σ-Protocols for Disjunctions. In: EUROCRYPT 2022. pp. 458–487. Springer (2022)
19. Goldwasser, S.: How to play any mental game, or a completeness theorem for protocols with an honest majority. In: STOC 1987. pp. 218–229 (1987)
20. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**(1), 186–208 (1989)
21. 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)
22. McCorry, P., Shahandashti, S.F., Hao, F.: A smart contract for boardroom voting with maximum voter privacy. In: FC 2017. pp. 357–375. Springer (2017)
23. Okamoto, T.: An efficient divisible electronic cash scheme. In: CRYPTO 1995. pp. 438–451. Springer (1995)
24. Okamoto, T., Takashima, K.: Hierarchical predicate encryption for inner-products. In: ASIACRYPT 2009. pp. 214–231. Springer (2009)
25. Okamoto, T., Takashima, K.: Adaptively attribute-hiding (hierarchical) inner product encryption. In: EUROCRYPT 2012. pp. 591–608. Springer (2012)
26. O'Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive (2010)
27. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: CRYPTO 1991. pp. 129–140. Springer (1991)
28. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: FOCS 1999. pp. 543–553. IEEE Computer Society (1999)
29. Sasson, E.B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: S&P 2014. pp. 459–474. IEEE (2014)
30. Schnorr, C.: Efficient identification and signatures for smart cards. In: CRYPTO 1989. pp. 239–252. Springer (1989)
31. Tan, S.Y., Groß, T.: Monipoly—an expressive q-SDH-based anonymous attribute-based credential system. In: ASIACRYPT 2020. pp. 498–526. Springer (2020)
32. Wee, H.: Dual system encryption via predicate encodings. In: TCC 2014. pp. 616–637. Springer (2014)

33. Wee, H.: Attribute-hiding predicate encryption in bilinear groups, revisited. In: TCC 2017. pp. 206–233. Springer (2017)
34. Zhang, Y.: Introducing zkEVM (2022), https://scroll.io/blog/zkEVM

A Preliminary: NIZK and Sigma protocols

A.1 NIZK

We recall the definition of NIZK as follows.

Definition 8. (NIZK). *Let $\mathcal{L}$ be an* NP *language associated with an* NP *relation $\mathcal{R}$. A* non-interactive zero-knowledge proof (NIZK) *for $\mathcal{L}$ consists of a tuple of three efficient algorithms* $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$*:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$*: On input a security parameter, the algorithm outputs a common reference string* crs*.*
- $\mathsf{Prove}(\mathsf{crs}, x, w) \to \pi$*: On input a common reference string* crs*, a statement x and a witness w, the proving algorithm outputs a proof π.*
- $\mathsf{Verify}(\mathsf{crs}, x, \pi) \to b \in \{0,1\}$*: On input a common reference string* crs*, a statement x and a proof π, the verification algorithm outputs a bit $b \in \{0,1\}$.*

Moreover, the algorithms are required to satisfy the following properties:

1. ***Completeness.*** *For any $(x, w) \in \mathcal{R}$, we have that*

$$\Pr\left[\begin{matrix}\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)\end{matrix} : \mathsf{Verify}(\mathsf{crs}, x, \pi) = 1\right] = 1.$$

2. ***Soundness.*** *Here we present two variants of soundness:*
   (a) *Non-adaptive soundness: for any $x \notin \mathcal{L}$ and any computationally unbounded adversary $\mathcal{A}$,*

$$\Pr\left[\begin{matrix}\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ \pi \leftarrow \mathcal{A}(\mathsf{crs}, x)\end{matrix} : \mathsf{Verify}(\mathsf{crs}, x, \pi) = 1\right] \leq \mathsf{negl}(\lambda).$$

   (b) *Adaptive soundness: for any computationally unbounded adversary $\mathcal{A}$,*

$$\Pr\left[\begin{matrix}\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs})\end{matrix} : \begin{matrix}x \notin \mathcal{L} \\ \wedge\ \mathsf{Verify}(\mathsf{crs}, x, \pi) = 1\end{matrix}\right] \leq \mathsf{negl}(\lambda).$$

3. ***Zero knowledge.*** *We also consider two variants of zero knowledge:* single-theorem *zero knowledge and* multi-theorem *zero knowledge.*
   (a) *Single-theorem zero knowledge: for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a PPT simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ such that for any polynomial l,*

$$|\Pr[\mathsf{ExpReal}^{\text{sin-zk}}_{\mathsf{NIZK},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{ExpIdeal}^{\text{sin-zk}}_{\mathsf{NIZK},\mathcal{A},\mathsf{Sim}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda),$$

   where $\mathsf{ExpReal}^{\text{sin-zk}}_{\mathsf{NIZK},\mathcal{A}}(\lambda)$ and $\mathsf{ExpIdeal}^{\text{sin-zk}}_{\mathsf{NIZK},\mathcal{A},\mathsf{Sim}}(\lambda)$ are defined in Fig. 12.

$\underline{\mathsf{ExpReal}^{\text{sin-zk}}_{\mathsf{NIZK},\mathcal{A}}(\lambda)}$:	$\underline{\mathsf{ExpIdeal}^{\text{sin-zk}}_{\mathsf{NIZK},\mathcal{A},\mathsf{Sim}}(\lambda)}$:
$\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$	$(\mathsf{crs}, st^{\mathsf{Sim}}) \leftarrow \mathsf{Sim}_1(1^\lambda)$
$((x,w), st^{\mathcal{A}}) \leftarrow \mathcal{A}_1(\mathsf{crs})$	$((x,w), st^{\mathcal{A}}) \leftarrow \mathcal{A}_1(\mathsf{crs})$
s.t. $(x,w) \in \mathcal{R}$	s.t. $(x,w) \in \mathcal{R}$
$\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$	$\pi \leftarrow \mathsf{Sim}_2(\mathsf{crs}, x, st^{\mathsf{Sim}})$
Return $b \leftarrow \mathcal{A}_2(\pi, st^{\mathcal{A}})$	Return $b \leftarrow \mathcal{A}_2(\pi, st^{\mathcal{A}})$

Fig. 12: Games for defining single-theorem zero knowledge for NIZK

$\underline{\mathsf{ExpReal}^{\text{mul-zk}}_{\mathsf{NIZK},\mathcal{A}}(\lambda)}$:	$\underline{\mathsf{ExpIdeal}^{\text{mul-zk}}_{\mathsf{NIZK},\mathcal{A},\mathsf{Sim}}(\lambda)}$:
$\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$	$(\mathsf{crs}, st^{\mathsf{Sim}}) \leftarrow \mathsf{Sim}_1(1^\lambda)$
$b \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Prove}}(\cdot,\cdot)}(\mathsf{crs})$	$b \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Prove}}(\cdot,\cdot)}(\mathsf{crs})$
Return b	Return b
$\underline{\mathcal{O}^{\mathsf{Prove}}(x,w)}$:	$\underline{\mathcal{O}^{\mathsf{Prove}}(x,w)}$:
If $(x,w) \notin \mathcal{R}$: Return $\perp$	If $(x,w) \notin \mathcal{R}$: Return $\perp$
$\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$	$(\pi, st^{\mathsf{Sim}}) \leftarrow \mathsf{Sim}_2(\mathsf{crs}, x, st^{\mathsf{Sim}})$
Return π	Return π

Fig. 13: Games for defining multi-theorem zero knowledge for NIZK

(b) Multi-theorem zero knowledge: for any PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ such that for any polynomial l,

$$|\Pr[\mathsf{ExpReal}^{\text{mul-zk}}_{\mathsf{NIZK},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{ExpIdeal}^{\text{mul-zk}}_{\mathsf{NIZK},\mathcal{A},\mathsf{Sim}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda),$$

where $\mathsf{ExpReal}^{\text{mul-zk}}_{\mathsf{NIZK},\mathcal{A}}(\lambda)$ and $\mathsf{ExpIdeal}^{\text{mul-zk}}_{\mathsf{NIZK},\mathcal{A},\mathsf{Sim}}(\lambda)$ are defined in Fig. 13.

A.2 Sigma protocols

Sigma protocols are widely-used interactive protocols for proof of knowledge. A Sigma protocol $\Sigma = (\mathcal{P}, \mathcal{V})$ for polynomial-time relation $\mathcal{R}$ consists of two efficient interactive protocol algorithms $(\mathcal{P}, \mathcal{V})$, where $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2)$ is the prover and $\mathcal{V} = (\mathcal{V}_1, \mathcal{V}_2)$ is the verifier, associated with a challenge space $\mathcal{CL}$. Specifically, for any $(x,w) \in \mathcal{R}$, the input of the prover (resp. verifier) is (x,w) (resp. x). The prover first computes $(\mathsf{a}, \mathsf{aux}) \leftarrow \mathcal{P}_1(x,w)$ and sends the commitment a to the verifier. The verifier (i.e., $\mathcal{V}_1$) returns a challenge $\mathsf{c} \leftarrow \mathcal{CL}$. Then the prover replies with $\mathsf{z} \leftarrow \mathcal{P}_2(\mathsf{a}, \mathsf{c}, x, w, \mathsf{aux})$. Receiving z, the verifier (i.e., $\mathcal{V}_2$) outputs $b \in \{0,1\}$. The tuple $(\mathsf{a}, \mathsf{c}, \mathsf{z})$ is called a *transcript*. We require that $\mathcal{V}$ does not make any random choices other than the selection of c. For any fixed $(\mathsf{a}, \mathsf{c}, \mathsf{z})$, if the final output of $\mathcal{V}(x)$ is 1, $(\mathsf{a}, \mathsf{c}, \mathsf{z})$ is called an *accepting transcript* for x. For simplicity, we denote by $\langle \mathcal{P}(w), \mathcal{V} \rangle(x)$ the final output of $\mathcal{V}$ when running the protocol $(\mathcal{P}, \mathcal{V})$ on common input x with $\mathcal{P}$ running on additional input w.

Completeness requires that for all $(x,w) \in \mathcal{R}$, when $\mathcal{P}(x,w)$ and $\mathcal{V}(x)$ interact with each other, the final output of $\mathcal{V}(x)$ is always 1.

The corresponding security notions are as follows.

Definition 9. (**Knowledge soundness**). *We say that a Sigma protocol* $(\mathcal{P}, \mathcal{V})$ *for* $\mathcal{R}$ *provides* knowledge soundness*, if there is an efficient deterministic algorithm* Ext *such that on input* $x \in \mathcal{X}$ *and two accepting transcripts* $(\mathsf{a}, \mathsf{c}, \mathsf{z}), (\mathsf{a}, \mathsf{c}', \mathsf{z}')$ *where* $\mathsf{c} \neq \mathsf{c}'$*,* Ext *always outputs a* $w \in \mathcal{W}$ *satisfying* $(x, w) \in \mathcal{R}$*.*

Definition 10. (**Special HVZK**). *We say that a Sigma protocol* $(\mathcal{P}, \mathcal{V})$ *for* $\mathcal{R}$ *with challenge space* $\mathcal{CL}$ *is* special honest verifier zero knowledge (special HVZK)*, if there is a PPT simulator* Sim *which takes* $(x, \mathsf{c}) \in \mathcal{X} \times \mathcal{CL}$ *as input and satisfies the following properties:*

(i) for all $(x, \mathsf{c}) \in \mathcal{X} \times \mathcal{CL}$*,* Sim *always outputs a pair* (a, z) *such that* $(\mathsf{a}, \mathsf{c}, \mathsf{z})$ *is an accepting transcript for* x*;*

(ii) for all $(x, w) \in \mathcal{R}$*, the tuple* $(\mathsf{a}, \mathsf{c}, \mathsf{z})$*, generated via* $\mathsf{c} \leftarrow \mathcal{CL}$ *and* $(\mathsf{a}, \mathsf{z}) \leftarrow \mathsf{Sim}(x, \mathsf{c})$*, has the same distribution as that of a transcript between* $\mathcal{P}(x, w)$ *and* $\mathcal{V}(x)$*.*

The Sigma protocols can be combined using "AND/OR"-proof construction and the combined protocol is also a Sigma protocol satisfying the above security properties [3]. In addition, applying the Fiat-Shamir transform [17] to a Sigma protocol, we can obtain a NIZK (with adaptive soundness).

A.3 Stackable Sigma protocols [18]

Here, we recall some definitions and theorems about stackable Sigma protocols, which are proposed in [18].

Definition 11. (Well-behaved simulator). *We say that* $\Sigma = (\mathcal{P}, \mathcal{V})$ *for an* NP *language* $\mathcal{L}$ *and associated relation* $\mathcal{R}$ *has a* well-behaved simulator*, if the simulator* Sim *defined for special HVZK has the following property:*

For any statement x *(for both* $x \in \mathcal{L}$ *and* $x \notin \mathcal{L}$*),*

$$\Pr[\mathsf{c} \leftarrow \mathcal{CL}; (\mathsf{a}, \mathsf{z}) \leftarrow \mathsf{Sim}(x, \mathsf{c}) : \mathcal{V}_2(x, \mathsf{a}, \mathsf{c}, \mathsf{z}) = 1] = 1$$

Definition 12. (EHVZK). *Let* $\Sigma = (\mathcal{P}, \mathcal{V})$ *be a Sigma protocol for the* NP *relation* $\mathcal{R}$*, with a well-behaved simulator. We say that* Σ *is* extended honest verifier zero knowledge (EHVZK)*, if there exists a polynomial time computable deterministic extended simulator* $\mathsf{Sim}_{\mathrm{EHVZK}}$*, such that for any* $(x, w) \in \mathcal{R}$ *and* $\mathsf{c} \in \mathcal{CL}$*, there exists an efficiently samplable distribution* $\mathcal{D}^{(\mathsf{z})}_{x,\mathsf{c}}$ *and the following distributions coincide:*

$$\{(\mathsf{a}, \mathsf{c}, \mathsf{z}) \mid r \leftarrow \mathcal{RS}; \mathsf{a} \leftarrow \mathcal{P}_1(x, w; r); \mathsf{z} \leftarrow \mathcal{P}_2(\mathsf{a}, \mathsf{c}, x, w, r)\},$$
$$\{(\mathsf{a}, \mathsf{c}, \mathsf{z}) \mid \mathsf{z} \leftarrow \mathcal{D}^{(\mathsf{z})}_{x,\mathsf{c}}; \mathsf{a} \leftarrow \mathsf{Sim}_{\mathrm{EHVZK}}(x, \mathsf{c}, \mathsf{z})\}.$$

Definition 13. (Recyclable third message). *Let* $\Sigma = (\mathcal{P}, \mathcal{V})$ *be a Sigma protocol for the* NP *relation* $\mathcal{R}$*, with a well-behaved simulator. We say that* Σ *has* recyclable third messages *if for each* $\mathsf{c} \in \mathcal{CL}$*, there exists an efficiently samplable distribution* $\mathcal{D}^{(\mathsf{z})}_{\mathsf{c}}$*, such that for all pairs* $(x, w) \in \mathcal{R}$*, it holds that:*

$$\mathcal{D}^{(\mathsf{z})}_{\mathsf{c}} \approx \{\mathsf{z} \mid r \leftarrow \mathcal{RS}; \mathsf{a} \leftarrow \mathcal{P}_1(x, w; r); \mathsf{z} \leftarrow \mathcal{P}_2(\mathsf{a}, \mathsf{c}, x, w, r)\}.$$

Definition 14. (Stackable Sigma protocol). *We say that a Sigma protocol* $\Sigma = (\mathcal{P}, \mathcal{V})$ *is* stackable*, if it is EHVZK and has recyclable third messages.*

In [18], the self stacking technique is discussed in cases where a relation is composed of multiple identical sub-relations. If the Sigma protocol for each sub-relation is stackable, then the compiled Sigma, which is obtained by applying the self stacking technique to the Sigma protocols for each sub-relation, is also stackable, as shown in the following theorem.

Theorem 8. (Self stacking [18, Theorem 5]). *Let* $\Pi = (\mathcal{P}, \mathcal{V})$ *be a stackable Sigma protocol as defined in Def. 14 for* NP *relation* $\mathcal{R} \subset \mathcal{X} \times \mathcal{W}$ *and let* $\mathsf{VCommit} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Com}, \mathsf{Equ}, \mathsf{Dec})$ *be a* 1*-out-of-*l *binding vector commitment scheme as defined in [18]. For any public parameter* $pp \leftarrow \mathsf{VCommit.Setup}(1^\lambda)$*, the compiled protocol* $\Pi' = (\mathcal{P}', \mathcal{V}')$ *using the self stacking technique is a stackable Sigma protocol for the relation* $\mathcal{R}' \subset \mathcal{X}^l \times ([l] \times \mathcal{W})$*, where* $((x_1, \ldots, x_l), (\alpha, w)) \in \mathcal{R}'$ *if and only if* $(x_\alpha, w) \in \mathcal{R}$*.*

According to [18, Lemma 1], Σ protocol for ψ-preimages is stackable. We recall the theorem as follows.

Theorem 9. (Σ protocol for ψ-preimages is stackable [18, Lemma 1]). *Let* $\mathfrak{G}_1$ *and* $\mathfrak{G}_2$ *be groups with group operations* $*_1$ *and* $*_2$ *respectively, and let* $\psi : \mathfrak{G}_1 \to \mathfrak{G}_2$ *be a one-way group-homomorphism. Recall the simple* Σ*-protocol (denoted as* Π_ψ*) of Cramer and Damgård [14] for the relation of preimages* $\mathcal{R}_\psi$ *(*$(x, w) \in \mathcal{R}_\psi$ *if and only if it holds that* $x = \psi(w)$*, where* $x \in \mathfrak{G}_1$ *and* $w \in \mathfrak{G}_2$*). The protocol works as follows:*

1. $\mathcal{P}_1(x, w; r)$*: The prover samples* $r \leftarrow \mathfrak{G}_1$ *and sends the image* $\mathsf{a} = \psi(r) \in \mathfrak{G}_2$ *to the verifier.*
2. $\mathcal{V}_1(\mathsf{a})$*: On receiving* a *from the prover, the verifier samples a challenge* c *and sends it to the prover.*
3. $\mathcal{P}_2(\mathsf{a}, \mathsf{c}, x, w; r)$*: The prover interprets* c *as an integer from a subset* $\mathcal{CL} \subseteq \mathcal{Z}$ *and replies with* $\mathsf{z} = w^{\mathsf{c}} \ *_1 \ r$*.*
4. $\mathcal{V}_2(x, \mathsf{a}, \mathsf{c}, \mathsf{z})$*: The verifier checks* $\psi(\mathsf{z}) \stackrel{?}{=} x^{\mathsf{c}} \ *_2 \ \mathsf{a}$*.*

Completeness follows since ψ *is a homomorphism:* $\psi(\mathsf{z}) = \psi(w^{\mathsf{c}} \ *_1 \ r) = \psi(w)^{\mathsf{c}} \ *_2 \ \psi(r) = x^{\mathsf{c}} \ *_2 \ \mathsf{a}$*. The knowledge soundness error is* $1/|\mathcal{CL}|$*. For any homomorphism* ψ*,* Π_ψ *is stackable.*

B Preliminary: functional encryption

We recall the definition of functional encryption here.

Definition 15. (Functional encryption). *A* functional encryption (FE) *scheme for function family* $\mathbb{F}$ *on message space* $\mathcal{M}$ *(both of which are implicitly depend on* λ*) consists of four PPT algorithms* $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$*.*

- $\mathsf{Setup}(1^\lambda) \to (mpk, msk)$*: The setup algorithm takes the security parameter* λ *as input, and outputs a master public key* mpk *and a master secret key* msk*.*
- $\mathsf{KGen}(mpk, msk, f) \to sk_f$*: The key generation algorithm takes the master public key* mpk*, the master secret key* msk *and a function* $f \in \mathbb{F}$ *as input, outputs a secret key* sk_f*.*
- $\mathsf{Enc}(mpk, m) \to c$*: The encryption algorithm takes the master public key* mpk *and a message* $m \in \mathcal{M}$ *as input, and outputs a ciphertext* c*.*
- $\mathsf{Dec}(c, sk_f) \to y$*: The decryption algorithm takes a ciphertext* c *and a secret key* sk_f *as input and outputs* y*.*

Correctness requires that for any (mpk, msk) *generated by* $\mathsf{Setup}(1^\lambda)$*, any* $f \in \mathbb{F}$ *and* $m \in \mathcal{M}$*,* $\Pr[\mathsf{Dec}(\mathsf{Enc}(mpk, m), \mathsf{KGen}(mpk, msk, f)) = f(m)] = 1$.

Here, we recall indistinguishability-based security (IND security) for FE.

$\underline{\mathsf{Exp}^{\mathrm{ind}}_{\mathsf{FE},\mathcal{A}}(\lambda)}$:
$(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda)$, $b \leftarrow \{0,1\}$, $W := \emptyset$, $Q := \emptyset$
$(m_0, m_1, st) \leftarrow \mathcal{A}_1^{\mathcal{O}^{\mathsf{KGen}}(\cdot)}(mpk)$
  s.t. $(|m_0| = |m_1|) \wedge (\forall f' \in Q, f'(m_0) = f'(m_1))$
$W := \{m_0, m_1\}$, $c \leftarrow \mathsf{Enc}(mpk, m_b)$, $b' \leftarrow \mathcal{A}_2^{\mathcal{O}^{\mathsf{KGen}}(\cdot)}(c, st)$
Return $(b = b')$
$\underline{\mathcal{O}^{\mathsf{KGen}}(f')}$:
If $W \neq \emptyset$:
  Parse $W = \{m_0, m_1\}$
  If $f'(m_0) \neq f'(m_1)$: Return $\perp$
$Q := Q \cup \{f'\}$
Return $sk_{f'} \leftarrow \mathsf{FE.KGen}(mpk, msk, f')$

Fig. 14: Game for defining IND-security of FE

Definition 16. (IND security for FE). *A FE scheme* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *is IND secure, if for any PPT adversary* $\mathcal{A}$*, the advantage* $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{FE},\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathrm{ind}}_{\mathsf{FE},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|$ *is negligible, where* $\mathsf{Exp}^{\mathrm{ind}}_{\mathsf{FE},\mathcal{A}}(\lambda)$ *is defined in Fig. 14.*

C Preliminary: inner-product encryption

We recall the definition of inner-product encryption (IPE).

Definition 17. (Inner-product encryption). *An* inner-product encryption (IPE) *scheme for a message space* $\mathcal{M}$ *consists of four algorithms* $\mathsf{IPE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$.

- $\mathsf{Setup}(1^\lambda, l) \to (pk, msk)$*: On input the security parameter* 1^λ *and the dimension* l *of the vector space, the setup algorithm outputs a public key* pk *and a master secret key* msk*.*
- $\mathsf{KGen}(msk, \mathbf{x}) \to sk_{\mathbf{x}}$*: On input* msk *and a vector* $\mathbf{x}$*, the key generation algorithm outputs a secret key* $sk_{\mathbf{x}}$ *for* $\mathbf{x}$*.*
- $\mathsf{Enc}(pk, \mathbf{y}, m) \to c_{\mathbf{y}}$*: On input* pk*, a vector* $\mathbf{y}$ *and a message* $m \in \mathcal{M}$*, the encryption algorithm outputs a ciphertext* $c_{\mathbf{y}}$ *for* $\mathbf{y}$*.*
- $\mathsf{Dec}(c_{\mathbf{y}}, sk_{\mathbf{x}}) \to m$*: On input a ciphertext* $c_{\mathbf{y}}$ *for* $\mathbf{y}$ *and a secret key* $sk_{\mathbf{x}}$ *for* $\mathbf{x}$ *as input, the decryption algorithm outputs a* m*.*

Correctness requires that for all $m \in \mathcal{M}$ *and all vectors* $\mathbf{x}, \mathbf{y}$ *satisfying* $\langle \mathbf{x}, \mathbf{y} \rangle = 0$*, it holds that:*

$$\Pr\left[\begin{matrix}(pk, msk) \leftarrow \mathsf{Setup}(1^\lambda, l) \\ sk_{\mathbf{x}} \leftarrow \mathsf{KGen}(msk, \mathbf{x})\end{matrix} : \mathsf{Dec}(\mathsf{Enc}(pk, \mathbf{y}, m), sk_{\mathbf{x}}) = m\right] = 1.$$

We recall adaptive security and fully attribute-hiding property for IPE [25].

Definition 18. (Adaptive security and fully attribute-hiding property for IPE). *An IPE scheme* $\mathsf{IPE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *for message space* $\mathcal{M}$ *and dimension* l *is* adaptively secure and fully attribute-hiding*, if for any PPT adversary* $\mathcal{A}$*, the advantage* $\mathbf{Adv}^{\text{a-ah}}_{\mathsf{IPE},\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\text{a-ah}}_{\mathsf{IPE},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|$ *is negligible, where* $\mathsf{Exp}^{\text{a-ah}}_{\mathsf{IPE},\mathcal{A}}(\lambda)$ *is defined in Fig. 15.*

D Preliminary: commitment

Definition 19. (Commitment). *A* commitment *scheme with message space* $\mathcal{M}$ *contains three PPT algorithms* $\mathsf{Commit} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Dec})$*:*

- $\mathsf{Setup}(1^\lambda) \to pp$*: The setup algorithm takes the security parameter* 1^λ *as input and output a public parameter* pp*.*
- $\mathsf{Com}(pp, m; r_{\text{com}}) \to \mathsf{com}$*: The commitment algorithm takes as input the public parameter* pp *and* $m \in \mathcal{M}$*, with an inner randomized input* r_{com}*, and outputs a commitment* com*.*
- $\mathsf{Dec}(pp, \mathsf{com}, r_{\text{com}}, m) \to b$*: The decommitment algorithm takes as input the public parameter* pp*, a commitment* com*, a decommitment* r_{com} *and a message* $m \in \mathcal{M}$*, and outputs a bit* $b \in \{0, 1\}$ *depending on whether* m *is the committed message of* com*.*

A commitment scheme enjoys the following properties:

– **Correctness.** *For all* $m \in \mathcal{M}$ *and all* $r_{\text{com}} \in \mathcal{RS}_{\mathsf{Com}}$*, it holds that*

$$\Pr[pp \leftarrow \mathsf{Setup}(1^\lambda), \mathsf{com} \leftarrow \mathsf{Com}(pp, m; r_{\text{com}}) : \mathsf{Dec}(pp, \mathsf{com}, r_{\text{com}}, m) = 1] = 1.$$

– **Binding.** *For any PPT adversary* $\mathcal{A}$*,*

$$\Pr\left[\begin{matrix}pp \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{com}, m, r_{\text{com}}, m', r'_{\text{com}}) \leftarrow \mathcal{A}(pp)\end{matrix} : \begin{matrix}m \neq m' \\ \wedge\ \mathsf{Dec}(pp, \mathsf{com}, r_{\text{com}}, m) = 1 \\ \wedge\ \mathsf{Dec}(pp, \mathsf{com}, r'_{\text{com}}, m') = 1\end{matrix}\right] \leq \mathsf{negl}(\lambda).$$

$\underline{\mathsf{Exp}^{\text{a-ah}}_{\mathsf{IPE},\mathcal{A}}(\lambda)}$:
$(pk, msk) \leftarrow \mathsf{Setup}(1^\lambda, l)$, $b \leftarrow \{0,1\}$, $C_{\text{y,m}} := \emptyset$, $U_{\text{x}} := \emptyset$
$(\mathbf{y}_0, \mathbf{y}_1, m_0, m_1, st) \leftarrow \mathcal{A}_1^{\mathcal{O}^{\mathsf{KGen}}(\cdot)}(pk, l)$
s.t. if $m_0 \neq m_1$, $(\langle \mathbf{x}, \mathbf{y}_0 \rangle \neq 0) \wedge (\langle \mathbf{x}, \mathbf{y}_1 \rangle \neq 0)$ for all $\mathbf{x} \in U_{\text{x}}$; and
if $m_0 = m_1$, $((\langle \mathbf{x}, \mathbf{y}_0 \rangle \neq 0) \wedge (\langle \mathbf{x}, \mathbf{y}_1 \rangle \neq 0)) \vee (\langle \mathbf{x}, \mathbf{y}_0 \rangle = \langle \mathbf{x}, \mathbf{y}_1 \rangle = 0)$ for all $\mathbf{x} \in U_{\text{x}}$
$C_{\text{y,m}} := \{(\mathbf{y}_0, \mathbf{y}_1, m_0, m_1)\}$, $c \leftarrow \mathsf{Enc}(pk, \mathbf{y}_b, m_b)$
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}^{\mathsf{KGen}}(\cdot)}(c, st)$
Return $(b = b')$

$\underline{\mathcal{O}^{\mathsf{KGen}}(\mathbf{x})}$:
If $C_{\text{y,m}} \neq \emptyset$:
Parse $C_{\text{y,m}} = \{(\mathbf{y}_0, \mathbf{y}_1, m_0, m_1)\}$
If $m_0 \neq m_1$:
If $(\langle \mathbf{x}, \mathbf{y}_0 \rangle = 0) \vee (\langle \mathbf{x}, \mathbf{y}_1 \rangle = 0)$: Return $\perp$
If $m_0 = m_1$:
If $((\langle \mathbf{x}, \mathbf{y}_0 \rangle \neq 0) \wedge (\langle \mathbf{x}, \mathbf{y}_1 \rangle = 0)) \vee ((\langle \mathbf{x}, \mathbf{y}_0 \rangle = 0) \wedge (\langle \mathbf{x}, \mathbf{y}_1 \rangle \neq 0))$: Return $\perp$
$U_{\text{x}} := U_{\text{x}} \cup \{\mathbf{x}\}$
Return $sk_{\mathbf{x}} \leftarrow \mathsf{KGen}(msk, \mathbf{x})$

Fig. 15: Game for defining adaptive security and fully attribute-hiding property for IPE

– **Hiding.** *For any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,

$$\left| \Pr \left[\begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda), b \leftarrow \{0,1\} \\ (m_0, m_1, st) \leftarrow \mathcal{A}_1(pp) \\ \mathsf{com} \leftarrow \mathsf{Com}(pp, m_b) \\ b' \leftarrow \mathcal{A}_2(\mathsf{com}, st) \end{array} : b' = b \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

If $\mathcal{A}$ *is unbounded and* $\mathsf{negl}(\lambda)$ *is fixed to be* 0*, we say that the scheme* Commit *is* perfect hiding.

We say that Commit *supports Sigma protocols*, if there exists an efficient Sigma protocol for the following relation:

$$\mathcal{R}_{\text{com}} = \{(\mathsf{com}, (r_{\text{com}}, m)) : \mathsf{com} = \mathsf{Com}(pp, m; r_{\text{com}})\}.$$

Note that Perdersen commitment [27] can be proved by Okamoto's Sigma protocol [23]. Thus, the Pedersen commitment satisfies the above requirement.

E Proof of Theorem 1

Proof (of Theorem 1). It is easy to see that Π satisfies the completeness and soundness of NIZK. So we mainly discuss zero knowledge property here. Specifically, we focus on single-theorem zero knowledge of NIZK, the definition of which can be found in Appendix A.1.

$\underline{\mathcal{B}_1(\mathsf{crs}, pk_1)}$:	$\underline{\mathcal{B}_2(st_1^{\mathcal{B}} = (\mathsf{crs}, pk_1))}$:
$U_{\mathtt{cor}} := \emptyset$, $st_1^{\mathcal{B}} := (\mathsf{crs}, pk_1)$	$\mathsf{crs}^{\mathrm{zk}} := (\mathsf{crs}, pk_1)$, $w' \leftarrow \mathcal{W}$, $\tau_{\mathrm{p}} \leftarrow \mathcal{T}$
Return $(U_{\mathtt{cor}}, st_1^{\mathcal{B}})$	$((x, w), st^{\mathcal{A}}) \leftarrow \mathcal{A}_1^{\mathrm{zk}}(\mathsf{crs}^{\mathrm{zk}})$
	$st_2^{\mathcal{B}} := st^{\mathcal{A}}$
$\underline{\mathcal{B}_3(\pi, st_2^{\mathcal{B}} = st^{\mathcal{A}})}$:	Return $(1, \tau_{\mathrm{p}}, x, w, w', st_2^{\mathcal{B}})$
$b \leftarrow \mathcal{A}_2^{\mathrm{zk}}(\pi, st^{\mathcal{A}})$	
Return b	

$\underline{\mathsf{Sim}_1^{\mathrm{nizk}}(1^\lambda)}$:	$\underline{\mathsf{Sim}_2^{\mathrm{nizk}}(\mathsf{crs}^{\mathrm{zk}}, x, st^{\mathsf{Sim}})}$:
$(\mathsf{crs}, st^{\mathsf{Sim}}) \leftarrow \mathsf{Sim}_1^{\mathrm{fnizk}}(1^\lambda)$	Parse $\mathsf{crs}^{\mathrm{zk}} = (\mathsf{crs}, pk)$
$(pk, msk) \leftarrow \mathsf{UKGen}(\mathsf{crs})$	$w' \leftarrow \mathcal{W}$, $\tau_{\mathrm{p}} \leftarrow \mathcal{T}$
$\mathsf{crs}^{\mathrm{zk}} := (\mathsf{crs}, pk)$	$\pi \leftarrow \mathsf{Sim}_2^{\mathrm{fnizk}}(\mathsf{crs}, pk, \tau_{\mathrm{p}}, x, w', st^{\mathsf{Sim}})$
Return $(\mathsf{crs}^{\mathrm{zk}}, st^{\mathsf{Sim}})$	Return π

Fig. 16: $\mathcal{B}$ and $\mathsf{Sim}^{\mathrm{zk}}$ in the proof of Theorem 1

Let $\mathcal{W}$ denote the witness space of $\mathcal{R}$.

For any PPT adversary $\mathcal{A}^{\mathrm{zk}} = (\mathcal{A}_1^{\mathrm{zk}}, \mathcal{A}_2^{\mathrm{zk}})$ attacking single-theorem zero knowledge property of Π, consider a PPT adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ attacking zero knowledge property of fNIZK (for $n = 1$) as shown in Fig. 16. Note that $\mathcal{B}$ does not make any secret key generation queries. So we derive that

$$\Pr[\mathsf{ExpReal}^{\mathrm{zk}}_{\mathsf{fNIZK},\mathcal{B},1}(\lambda) = 1] = \Pr[\mathsf{ExpReal}^{\text{sin-zk}}_{\mathsf{NIZK},\mathcal{A}^{\mathrm{zk}}}(\lambda) = 1]. \tag{2}$$

Zero knowledge property of fNIZK guarantees that there exists a PPT simulator $\mathsf{Sim}^{\mathrm{fnizk}} = (\mathsf{Sim}_1^{\mathrm{fnizk}}, \mathsf{Sim}_2^{\mathrm{fnizk}})$, such that

$$|\Pr[\mathsf{ExpReal}^{\mathrm{zk}}_{\mathsf{fNIZK},\mathcal{B},1}(\lambda) = 1] - \Pr[\mathsf{ExpIdeal}^{\mathrm{zk}}_{\mathsf{fNIZK},\mathcal{B},\mathsf{Sim}^{\mathrm{fnizk}},1}(\lambda) = 1]| \leq \mathsf{negl}(\lambda). \tag{3}$$

Based on $\mathsf{Sim}^{\mathrm{fnizk}}$, we construct a simulator $\mathsf{Sim}^{\mathrm{nizk}} = (\mathsf{Sim}_1^{\mathrm{nizk}}, \mathsf{Sim}_2^{\mathrm{nizk}})$ for zero knowledge as shown in Fig. 16. Obviously, we have that

$$\Pr[\mathsf{ExpIdeal}^{\text{sin-zk}}_{\mathsf{NIZK},\mathcal{A}^{\mathrm{zk}},\mathsf{Sim}^{\mathrm{nizk}}}(\lambda) = 1] = \Pr[\mathsf{ExpIdeal}^{\mathrm{zk}}_{\mathsf{fNIZK},\mathcal{B},\mathsf{Sim}^{\mathrm{fnizk}},1}(\lambda) = 1]. \tag{4}$$

Combining Eqs. (2)-(4), we derive that

$$|\Pr[\mathsf{ExpReal}^{\text{sin-zk}}_{\mathsf{NIZK},\mathcal{A}^{\mathrm{zk}}}(\lambda) = 1] - \Pr[\mathsf{ExpIdeal}^{\text{sin-zk}}_{\mathsf{NIZK},\mathcal{A}^{\mathrm{zk}},\mathsf{Sim}^{\mathrm{nizk}}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda),$$

concluding this proof. □

F Proof of Theorem 2

Proof (of Theorem 2). We show the proof of zero knowledge with a sequence of games.

Game $\mathbf{G}_0$: For any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, let $\mathbf{G}_0$ be the real game $\mathsf{ExpReal}^{\mathrm{zk}}_{\mathsf{fNIZK},\mathcal{A},n}(\lambda)$. Specifically, the challenger firstly generates $\mathsf{crs} = (\mathsf{crs}_{\mathcal{R}_{\mathrm{ct}}},$

$\mathsf{crs}_{\mathcal{R}_\mathrm{k}}$) with algorithms $\mathsf{NIZK}_{\mathcal{R}_\mathrm{ct}}.\mathsf{Setup}$ and $\mathsf{NIZK}_{\mathcal{R}_\mathrm{k}}.\mathsf{Setup}$, generates $(pk_i = mpk_i, sk_i = msk_i)_{i\in[n]}$ with algorithm $\mathsf{FE.Setup}$, and initiates two sets $W := \emptyset$ and $Q := \emptyset$. Then, the challenger sends $(\mathsf{crs}, (pk)_{i\in[n]})$ to $\mathcal{A}_1$. Receiving $U_{\mathtt{cor}} \subset [n]$ from $\mathcal{A}_1$, the challenger sends $(sk_i)_{i\in U_{\mathtt{cor}}}$ to $\mathcal{A}_2$, and answers $\mathcal{A}_2$'s oracle queries with $(sk_i)_{i\in[n]}$ (i.e., for each of $\mathcal{A}_2$'s query $(i', f', \tau'_\mathtt{f})$, set $Q := Q\cup\{(i', f', \tau'_\mathtt{f})\}$, computes $sk_{i',\widehat{f}_{f',\tau'_\mathtt{f}}} \leftarrow \mathsf{FE.KGen}(mpk_{i'}, msk_{i'}, \widehat{f}_{f',\tau'_\mathtt{f}}; r_\mathrm{kg})$ (where $r_\mathrm{kg} \leftarrow \mathcal{RS}_{\mathsf{FE.KGen}}$) and $\Pi_{\mathcal{R}_\mathrm{k}} \leftarrow \mathsf{NIZK}_{\mathcal{R}_\mathrm{k}}.\mathsf{Prove}(\mathsf{crs}_{\mathcal{R}_\mathrm{k}}, (pk_{i'}, \widehat{f}_{f',\tau'_\mathtt{f}}, sk_{i',\widehat{f}_{f',\tau'_\mathtt{f}}}), (sk_{i'}, r_\mathrm{kg}))$, and returns $sk_{i',f',\tau'_\mathtt{f}} = (sk_{i',\widehat{f}_{f',\tau'_\mathtt{f}}}, \Pi_{\mathcal{R}_\mathrm{k}})$).

Receiving the challenge tuple $(i^*, \tau_\mathtt{p}, x, w, w')$ from $\mathcal{A}_2$ (the challenge tuple satisfies that (i) $i^* \notin U_{\mathtt{cor}}$, (ii) $(x, w) \in \mathcal{R}$, and (iii) for all existing query pairs $(i', f', \tau'_\mathtt{f})$ in Q, if $i' = i^*$ and $\mathsf{P}(\tau_\mathtt{p}, \tau'_\mathtt{f}) = 1$, then $f'(w) = f'(w')$), the challenger sets $W := \{i^*, \tau_\mathtt{p}, w, w'\}$. Then, it samples $r_\mathrm{enc} \leftarrow \mathcal{RS}_{\mathsf{FE.Enc}}$, and computes $c = \mathsf{FE.Enc}(mpk_{i^*}, (w, \tau_\mathtt{p}); r_\mathrm{enc})$ and $\Pi_{\mathcal{R}_\mathrm{ct}} \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\tau_\mathtt{p}, x, mpk_{i^*}, c), (w, r_\mathrm{enc}))$. Subsequently, it sends $\pi = (\tau_\mathtt{p}, \Pi_{\mathcal{R}_\mathrm{ct}}, c)$ to $\mathcal{A}_3$, and then answers $\mathcal{A}_3$'s oracle queries as follows:

- $\underline{\mathcal{O}^{\mathsf{FKGen}}(i', f', \tau'_\mathtt{f})}$: The challenger parses $W = \{i^*, \tau_\mathtt{p}, w, w'\}$, and proceeds as below:
  - If $(i' = i^*) \wedge \mathsf{P}(\tau_\mathtt{p}, \tau'_\mathtt{f}) = 1 \wedge f'(w) \neq f'(w')$, return $\perp$;
  - Else, set $Q := Q\cup\{(i', f', \tau'_\mathtt{f})\}$, computes $sk_{i',\widehat{f}_{f',\tau'_\mathtt{f}}} \leftarrow \mathsf{FE.KGen}(mpk_{i'}, msk_{i'}, \widehat{f}_{f',\tau'_\mathtt{f}}; r_\mathrm{kg})$ (where $r_\mathrm{kg} \leftarrow \mathcal{RS}_{\mathsf{FE.KGen}}$) and $\Pi_{\mathcal{R}_\mathrm{k}} \leftarrow \mathsf{NIZK}_{\mathcal{R}_\mathrm{k}}.\mathsf{Prove}(\mathsf{crs}_{\mathcal{R}_\mathrm{k}}, (pk_{i'}, \widehat{f}_{f',\tau'_\mathtt{f}}, sk_{i',\widehat{f}_{f',\tau'_\mathtt{f}}}), (sk_{i'}, r_\mathrm{kg}))$, and returns $sk_{i',f',\tau'_\mathtt{f}} = (sk_{i',\widehat{f}_{f',\tau'_\mathtt{f}}}, \Pi_{\mathcal{R}_\mathrm{k}})$.

  Finally, the challenger returns $\mathcal{A}_3$'s final output b as its own final output.

Since $\mathbf{G}_0 = \mathsf{ExpReal}^{\mathsf{zk}}_{\mathsf{fNIZK},\mathcal{A},n}(\lambda)$, we derive that

$$\Pr[\mathbf{G}_0 = 1] = \Pr[\mathsf{ExpReal}^{\mathsf{zk}}_{\mathsf{fNIZK},\mathcal{A},n}(\lambda) = 1]. \tag{5}$$

Game $\mathbf{G}_1$: This game is the same as $\mathbf{G}_0$, except that $\mathsf{crs}_{\mathcal{R}_\mathrm{ct}}$ and $\Pi_{\mathcal{R}_\mathrm{ct}}$ are both generated by the corresponding simulator of $\mathsf{NIZK}_{\mathcal{R}_\mathrm{ct}}$. Specifically, in this game, (i) $\mathsf{crs}_{\mathcal{R}_\mathrm{ct}}$ is generated via $(\mathsf{crs}_{\mathcal{R}_\mathrm{ct}}, st^{\mathsf{Sim}}) \leftarrow \mathsf{NIZK}_{\mathcal{R}_\mathrm{ct}}.\mathsf{Sim}_1(1^\lambda)$, and (ii) $\Pi_{\mathcal{R}_\mathrm{ct}}$ is generated via $\Pi_{\mathcal{R}_\mathrm{ct}} \leftarrow \mathsf{NIZK}_{\mathcal{R}_\mathrm{ct}}.\mathsf{Sim}_2(\mathsf{crs}_{\mathcal{R}_\mathrm{ct}}, (\tau_\mathtt{p}, x, mpk_{i^*}, c), st^{\mathsf{Sim}})$, where $\mathsf{NIZK}_{\mathcal{R}_\mathrm{ct}}.\mathsf{Sim} = (\mathsf{NIZK}_{\mathcal{R}_\mathrm{ct}}.\mathsf{Sim}_1, \mathsf{NIZK}_{\mathcal{R}_\mathrm{ct}}.\mathsf{Sim}_2)$ is the corresponding simulator of $\mathsf{NIZK}_{\mathcal{R}_\mathrm{ct}}$.

It is obvious that

$$\begin{aligned}
&|\Pr[\mathbf{G}_1 = 1] - \Pr[\mathbf{G}_0 = 1]| \\
&= |\Pr[\mathsf{ExpIdeal}^{\text{sin-zk}}_{\mathsf{NIZK}_{\mathcal{R}_\mathrm{ct}},\mathcal{A}',\mathsf{NIZK}_{\mathcal{R}_\mathrm{ct}}.\mathsf{Sim}}(\lambda) = 1] - \Pr[\mathsf{ExpReal}^{\text{sin-zk}}_{\mathsf{NIZK}_{\mathcal{R}_\mathrm{ct}},\mathcal{A}'}(\lambda) = 1]| \\
&\leq \mathsf{negl}(\lambda),
\end{aligned} \tag{6}$$

for some PPT adversary $\mathcal{A}'$ (i.e., $\mathcal{A}'$ is the PPT adversary attacking single-theorem zero knowledge of $\mathsf{NIZK}_{\mathcal{R}_\mathrm{ct}}$ based on $\mathcal{A}$).

Game $\mathbf{G}_2$: This game is the same as $\mathbf{G}_1$, except that $\mathsf{crs}_{\mathcal{R}_\mathrm{k}}$ and all the $\Pi'_{\mathcal{R}_\mathrm{k}}$ in the responses of the secret key generation oracle are generated by the corresponding simulator of $\mathsf{NIZK}_{\mathcal{R}_\mathrm{k}}$. Specifically, in this game, (i) $\mathsf{crs}_{\mathcal{R}_\mathrm{k}}$ is generated via

$(\mathsf{crs}_{\mathcal{R}_\mathsf{k}}, st^{\mathsf{Sim}}) \leftarrow \mathsf{NIZK}_{\mathcal{R}_\mathsf{k}}.\mathsf{Sim}_1(1^\lambda)$, and (ii) for each secret key generation query $(i', f', \tau'_\mathsf{f})$, after generating $sk_{i',\widehat{f}_{f',\tau'_\mathsf{f}}} \leftarrow \mathsf{FE.KGen}(pk_{i'}, sk_{i'}, \widehat{f}_{f',\tau'_\mathsf{f}})$, the corresponding $\Pi'_{\mathcal{R}_\mathsf{k}}$ is generated via $\Pi'_{\mathcal{R}_\mathsf{k}} \leftarrow \mathsf{NIZK}_{\mathcal{R}_\mathsf{k}}.\mathsf{Sim}_2(\mathsf{crs}_{\mathcal{R}_\mathsf{k}}, (pk_{i'}, \widehat{f}_{f',\tau'_\mathsf{f}}, sk_{i',\widehat{f}_{f',\tau'_\mathsf{f}}}), st^{\mathsf{Sim}})$, where $\mathsf{NIZK}_{\mathcal{R}_\mathsf{k}}.\mathsf{Sim} = (\mathsf{NIZK}_{\mathcal{R}_\mathsf{k}}.\mathsf{Sim}_1, \mathsf{NIZK}_{\mathcal{R}_\mathsf{k}}.\mathsf{Sim}_2)$ is the corresponding simulator of $\mathsf{NIZK}_{\mathcal{R}_\mathsf{k}}$.

It is obvious that

$$\begin{aligned} &|\Pr[\mathbf{G}_2 = 1] - \Pr[\mathbf{G}_1 = 1]| \\ &= |\Pr[\mathsf{ExpIdeal}^{\text{mul-zk}}_{\mathsf{NIZK}_{\mathcal{R}_\mathsf{k}},\mathcal{A}'',\mathsf{NIZK}_{\mathcal{R}_\mathsf{k}}.\mathsf{Sim}}(\lambda) = 1] - \Pr[\mathsf{ExpReal}^{\text{mul-zk}}_{\mathsf{NIZK}_{\mathcal{R}_\mathsf{k}},\mathcal{A}''}(\lambda) = 1]| \\ &\leq \mathsf{negl}(\lambda), \end{aligned} \tag{7}$$

for some PPT adversary $\mathcal{A}''$ (i.e., $\mathcal{A}''$ is the PPT adversary attacking multi-theorem zero knowledge of $\mathsf{NIZK}_{\mathcal{R}_\mathsf{k}}$ based on $\mathcal{A}$).

Game $\mathbf{G}_3$: This game is the same as $\mathbf{G}_2$, except that c (which is obtained via encrypting w and τ_p in $\mathbf{G}_2$) is replaced with $c \leftarrow \mathsf{FE.Enc}(mpk_{i^*}, (w', \tau_\mathsf{p}))$.

Obviously,

$$|\Pr[\mathbf{G}_3 = 1] - \Pr[\mathbf{G}_2 = 1]| \leq 2n \cdot \mathbf{Adv}^{\text{ind}}_{\mathsf{FE},\mathcal{A}'''}(\lambda) \tag{8}$$

for some PPT adversary $\mathcal{A}'''$ attacking IND security of FE, where the security loss n is incurred due to the necessity of $\mathcal{A}'''$ to make a guess on the challenge index i^*.

A PPT simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ can be constructed as shown in Fig. 17.

$\mathsf{Sim}_1(1^\lambda)$:	$\mathsf{Sim}_2(\mathsf{crs}, pk_{i^*}, \tau_\mathsf{p}, x, w', st^{\mathsf{Sim}})$:
$(\mathsf{crs}_{\mathcal{R}_\mathsf{ct}}, st^{\mathsf{Sim}}_{\mathcal{R}_\mathsf{ct}}) \leftarrow \mathsf{NIZK}_{\mathcal{R}_\mathsf{ct}}.\mathsf{Sim}_1(1^\lambda)$	Parse $\mathsf{crs} = (\mathsf{crs}_{\mathcal{R}_\mathsf{ct}}, \mathsf{crs}_{\mathcal{R}_\mathsf{k}})$, $st^{\mathsf{Sim}} = (st^{\mathsf{Sim}}_{\mathcal{R}_\mathsf{ct}}, st^{\mathsf{Sim}}_{\mathcal{R}_\mathsf{k}})$
$(\mathsf{crs}_{\mathcal{R}_\mathsf{k}}, st^{\mathsf{Sim}}_{\mathcal{R}_\mathsf{k}}) \leftarrow \mathsf{NIZK}_{\mathcal{R}_\mathsf{k}}.\mathsf{Sim}_1(1^\lambda)$	$c \leftarrow \mathsf{FE.Enc}(pk_{i^*}, (w', \tau_\mathsf{p}))$
$\mathsf{crs} := (\mathsf{crs}_{\mathcal{R}_\mathsf{ct}}, \mathsf{crs}_{\mathcal{R}_\mathsf{k}})$	$\Pi_{\mathcal{R}_\mathsf{ct}} \leftarrow \mathsf{NIZK}_{\mathcal{R}_\mathsf{ct}}.\mathsf{Sim}_2(\mathsf{crs}_{\mathcal{R}_\mathsf{ct}}, (\tau_\mathsf{p}, x, pk_{i^*}, c), st^{\mathsf{Sim}}_{\mathcal{R}_\mathsf{ct}})$
$st^{\mathsf{Sim}} := (st^{\mathsf{Sim}}_{\mathcal{R}_\mathsf{ct}}, st^{\mathsf{Sim}}_{\mathcal{R}_\mathsf{k}})$	Return $\pi := (\tau_\mathsf{p}, \Pi_{\mathcal{R}_\mathsf{ct}}, c)$
Return $(\mathsf{crs}, st^{\mathsf{Sim}})$	

Fig. 17: Construction of simulator Sim in the proof of zero knowledge for fNIZK

Note that $\mathsf{ExpIdeal}^{\text{zk}}_{\mathsf{fNIZK},\mathcal{A},\mathsf{Sim},n}(\lambda)$ is totally the same as $\mathbf{G}_3$. So we have

$$\Pr[\mathsf{ExpIdeal}^{\text{zk}}_{\mathsf{fNIZK},\mathcal{A},\mathsf{Sim},n}(\lambda) = 1] = \Pr[\mathbf{G}_3 = 1]. \tag{9}$$

Hence, combining equations (5)-(9), we obtain

$$\begin{aligned} &|\Pr[\mathsf{ExpReal}^{\text{zk}}_{\mathsf{fNIZK},\mathcal{A},n}(\lambda) = 1] - \Pr[\mathsf{ExpIdeal}^{\text{zk}}_{\mathsf{fNIZK},\mathcal{A},\mathsf{Sim},n}(\lambda) = 1]| \\ &= |\Pr[\mathbf{G}_0 = 1] - \Pr[\mathbf{G}_3 = 1]| \leq \mathsf{negl}(\lambda) + 2n \cdot \mathbf{Adv}^{\text{ind}}_{\mathsf{FE},\mathcal{A}'''}(\lambda), \end{aligned}$$

which is also negligible. □

G Proof of Theorem 3

Proof (of Theorem 3). We show the proof of zero knowledge with a sequence of games.

Game $\mathbf{G}_0$: For any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, let $\mathbf{G}_0$ be the real game $\mathsf{ExpReal}^{\mathrm{zk}}_{\mathsf{fSMP},\mathcal{A},n}(\lambda)$. Specifically, the challenger firstly generates $\mathsf{crs} = (\mathsf{crs}_{\mathrm{nizk}}, pp)$ with algorithms $\mathsf{NIZK.Setup}$ and $\mathsf{Commit.Setup}$, and initiates two sets $W := \emptyset$ and $Q := \emptyset$. Then, the challenger computes $(pk_i, sk_i)_{i\in[n]}$ with algorithm $\mathsf{SME.Setup}$. Subsequently, the challenger sends $(\mathsf{crs}, (pk_i)_{i\in[n]})$ to $\mathcal{A}_1$. Receiving $U_{\mathtt{cor}} \subset [n]$ from $\mathcal{A}_1$, the challenger sends $(sk_i)_{i\in U_{\mathtt{cor}}}$ to $\mathcal{A}_2$, and answers $\mathcal{A}_2$'s oracle queries with $(sk_i)_{i\in[n]}$ (i.e., for each of $\mathcal{A}_2$'s query $(i', f', \tau'_{\mathtt{f}})$, where f' indicates a set $S_{f'} \subset [l]$, set $Q := Q \cup \{(i', f', \tau'_{\mathtt{f}})\}$, compute $sk_{i',S_{f'},\tau'_{\mathtt{f}}} \leftarrow \mathsf{SME.KGen}(pk_{i'}, sk_{i'}, S_{f'}, \tau'_{\mathtt{f}})$, and return $sk_{i',f',\tau'_{\mathtt{f}}} = sk_{i',S_{f'},\tau'_{\mathtt{f}}}$).

Receiving the challenge tuple $(i^*, \tau_{\mathtt{p}}, x, w_x, w'_x)$ from $\mathcal{A}_2$ (the challenge tuple satisfies that (i) $i^* \notin U_{\mathtt{cor}}$, (ii) $(x, w_x) \in \mathcal{R}$, and (iii) for all existing query pairs $(i', f', \tau'_{\mathtt{f}})$ in Q, if $\mathsf{P}(\tau_{\mathtt{p}}, \tau'_{\mathtt{f}}) = 1$, then $f'(w) = f'(w')$), the challenger sets $W := \{i^*, \tau_{\mathtt{p}}, w_x, w'_x\}$, and parses $x = (\mathsf{com}, \Phi)$, $w_x = (w, r_{\mathrm{com}})$ and $w'_x = (w', r'_{\mathrm{com}})$. Then, it samples $r_{\mathrm{enc}} \leftarrow \mathcal{RS}_{\mathsf{SME.Enc}}$, and computes $c = \mathsf{SME.Enc}(pk, \Phi, \tau_{\mathtt{p}}, w; r_{\mathrm{enc}})$ and $\pi_{\widetilde{\mathcal{R}_{\mathrm{sm}}}} \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathrm{nizk}}, (\tau_{\mathtt{p}}, \mathsf{com}, c, pk, \Phi), (w, r_{\mathrm{com}}, r_{\mathrm{enc}}))$. Subsequently, it sends $\pi = (\tau_{\mathtt{p}}, \pi_{\widetilde{\mathcal{R}_{\mathrm{sm}}}}, c)$ to $\mathcal{A}_3$, and then answers $\mathcal{A}_3$'s oracle queries as follows:

- $\underline{\mathcal{O}^{\mathsf{FKGen}}(i', f', \tau'_{\mathtt{f}})}$: The challenger parses $W = \{i^*, \tau_{\mathtt{p}}, (w, r_{\mathrm{com}}), (w', r'_{\mathrm{com}})\}$, and proceeds as below:
  - If $(i' = i^*) \wedge (\mathsf{P}(\tau_{\mathtt{p}}, \tau'_{\mathtt{f}}) = 1) \wedge (f'(w) \neq f'(w'))$, return $\perp$;
  - Else, set $Q := Q \cup \{(i', f', \tau'_{\mathtt{f}})\}$ and return $sk_{i',f',\tau'_{\mathtt{f}}} \leftarrow \mathsf{SME.KGen}(pk_{i'}, sk_{i'}, S_{f'}, \tau'_{\mathtt{f}})$, where $S_{f'} \subset [l]$ is the corresponding set indicated by f'.

Finally, the challenger returns $\mathcal{A}_3$'s final output b as its own final output. Since $\mathbf{G}_0 = \mathsf{ExpReal}^{\mathrm{zk}}_{\mathsf{fSMP},\mathcal{A},n}(\lambda)$, we derive that

$$\Pr[\mathbf{G}_0 = 1] = \Pr[\mathsf{ExpReal}^{\mathrm{zk}}_{\mathsf{fSMP},\mathcal{A},n}(\lambda) = 1]. \tag{10}$$

Game $\mathbf{G}_1$: This game is the same as $\mathbf{G}_0$, except that $\mathsf{crs}_{\mathrm{nizk}}$ and $\pi_{\widetilde{\mathcal{R}_{\mathrm{sm}}}}$ are both generated by the corresponding simulator of NIZK. Specifically, in this game, (i) $\mathsf{crs}_{\mathrm{nizk}}$ is generated via $(\mathsf{crs}_{\mathrm{nizk}}, st^{\mathsf{Sim}}) \leftarrow \mathsf{NIZK.Sim}_1(1^\lambda)$, and (ii) $\pi_{\widetilde{\mathcal{R}_{\mathrm{sm}}}}$ is generated via $\pi_{\widetilde{\mathcal{R}_{\mathrm{sm}}}} \leftarrow \mathsf{NIZK.Sim}_2(\mathsf{crs}_{\mathrm{nizk}}, (\tau_{\mathtt{p}}, \mathsf{com}, c, pk_{i^*}, \Phi), st^{\mathsf{Sim}})$, where $\mathsf{NIZK.Sim} = (\mathsf{NIZK.Sim}_1, \mathsf{NIZK.Sim}_2)$ is the corresponding simulator of NIZK.

It is obvious to see that

$$\begin{aligned} &|\Pr[\mathbf{G}_1 = 1] - \Pr[\mathbf{G}_0 = 1]| \\ &= |\Pr[\mathsf{ExpIdeal}^{\text{sin-zk}}_{\mathsf{NIZK},\mathcal{A}',\mathsf{NIZK.Sim}}(\lambda) = 1] - \Pr[\mathsf{ExpReal}^{\text{sin-zk}}_{\mathsf{NIZK},\mathcal{A}'}(\lambda) = 1]| \leq \mathsf{negl}(\lambda) \end{aligned} \tag{11}$$

for some PPT adversary $\mathcal{A}'$ (i.e., $\mathcal{A}'$ is the PPT adversary attacking single-theorem zero knowledge of NIZK based on $\mathcal{A}$).

Game $\mathbf{G}_2$: This game is the same as $\mathbf{G}_1$, except for the generation of c. Specifically, in this game, c is generated as follows:

– If $w' \notin \Phi$, sample $w_{\mathrm{r}} \leftarrow \Phi$, and then compute $c \leftarrow \mathsf{SME.Enc}(pk, \Phi, \tau_{\mathrm{p}}, w_{\mathrm{r}})$.
– If $w' \in \Phi$, compute $c \leftarrow \mathsf{SME.Enc}(pk, \Phi, \tau_{\mathrm{p}}, w')$.

We present the following lemma with a postponed proof.

Lemma 1. *There is a PPT adversary $\mathcal{A}''$ attacking the IND security of* SME*, such that*

$$|\Pr[\mathbf{G}_2 = 1] - \Pr[\mathbf{G}_1 = 1]| \leq 2n \cdot \mathbf{Adv}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}''}(\lambda). \tag{12}$$

A PPT simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ can be constructed as shown in Fig. 18.

$\underline{\mathsf{Sim}_1(1^\lambda)}$:	$\underline{\mathsf{Sim}_2(\mathsf{crs}, pk, \tau_{\mathrm{p}}, x, w'_x, st^{\mathsf{Sim}})}$:
$(\mathsf{crs}_{\mathrm{nizk}}, st^{\mathsf{Sim}}) \leftarrow \mathsf{NIZK.Sim}_1(1^\lambda)$	Parse $x = (\mathsf{com}, \Phi)$ and $w'_x = (w', r'_{\mathrm{com}})$
$pp \leftarrow \mathsf{Commit.Setup}(1^\lambda)$	If $w' \notin \Phi$: $w_{\mathrm{r}} \leftarrow \Phi$, $c \leftarrow \mathsf{SME.Enc}(pk, \Phi, \tau_{\mathrm{p}}, w_{\mathrm{r}})$
$\mathsf{crs} = (\mathsf{crs}_{\mathrm{nizk}}, pp)$	If $w' \in \Phi$: $c \leftarrow \mathsf{SME.Enc}(pk, \Phi, \tau_{\mathrm{p}}, w')$
Return $(\mathsf{crs}, st^{\mathsf{Sim}})$	$\pi_{\widetilde{\mathcal{R}_{\mathrm{sm}}}} \leftarrow \mathsf{NIZK.Sim}_2(\mathsf{crs}_{\mathrm{nizk}}, (\tau_{\mathrm{p}}, \mathsf{com}, c, pk, \Phi), st^{\mathsf{Sim}})$
	Return $\pi := (\tau_{\mathrm{p}}, \pi_{\widetilde{\mathcal{R}_{\mathrm{sm}}}}, c)$

Fig. 18: Construction of simulator Sim in the proof of zero knowledge for fSMP

Note that $\mathsf{ExpIdeal}^{\mathrm{zk}}_{\mathsf{fSMP},\mathcal{A},\mathsf{Sim},n}(\lambda)$ is totally the same as $\mathbf{G}_2$. So we have

$$\Pr[\mathsf{ExpIdeal}^{\mathrm{zk}}_{\mathsf{fSMP},\mathcal{A},\mathsf{Sim},n}(\lambda) = 1] = \Pr[\mathbf{G}_2 = 1]. \tag{13}$$

Hence, combining equations (10)-(13), we obtain

$$\begin{aligned} &|\Pr[\mathsf{ExpReal}^{\mathrm{zk}}_{\mathsf{fSMP},\mathcal{A},n}(\lambda) = 1] - \Pr[\mathsf{ExpIdeal}^{\mathrm{zk}}_{\mathsf{fSMP},\mathcal{A},\mathsf{Sim},n}(\lambda) = 1]| \\ = &|\Pr[\mathbf{G}_0 = 1] - \Pr[\mathbf{G}_2 = 1]| \leq \mathsf{negl}(\lambda) + 2n \cdot \mathbf{Adv}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}''}(\lambda), \end{aligned}$$

which is also negligible.

So what remains is to prove Lemma 1.

Proof (of Lemma 1). Consider the adversary $\mathcal{A}'' = (\mathcal{A}''_1, \mathcal{A}''_2)$ attacking the IND security of SME based on $\mathcal{A}$ as follows. $\mathcal{A}''$ simulates $\mathbf{G}_1$ for $\mathcal{A}$, where

– At the beginning, receiving pk from the challenger (in game $\mathsf{Exp}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}}(\lambda)$), $\mathcal{A}''_1$ uniformly samples $i^*_{\mathrm{gs}} \leftarrow [n]$ as the "guessed" i^*, sets $pk_{i^*_{\mathrm{gs}}} = pk$, and generates all the other $n-1$ key pairs and crs by itself. Then, $\mathcal{A}''_1$ sends $(\mathsf{crs}, (pk_i)_{i\in[n]})$ to $\mathcal{A}_1$. (If $\mathcal{A}''$ guess the wrong index, it aborts the simulation with a random bit is its final output.)
– $\mathcal{A}''_1$ answers $\mathcal{A}_2$'s secret functional key generation queries.
– Receiving the challenge tuple $(i^*, \tau_{\mathrm{p}}, x = (\mathsf{com}, \Phi), w_x = (w, r_{\mathrm{com}}), w'_x = (w', r'_{\mathrm{com}}))$ from $\mathcal{A}_2$, if $i^*_{\mathrm{gs}} \neq i^*$, $\mathcal{A}''$ aborts with a random bit as its own final output; otherwise, $\mathcal{A}''_1$ generates the challenge tuple for game $\mathsf{Exp}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}''}(\lambda)$ as follows:

- If $w' \notin \Phi$, $\mathcal{A}_1''$ samples $w_{\mathrm{r}} \leftarrow \Phi$, and returns $(\Phi, \tau_{\mathtt{p}}, w_0 = w, w_1 = w_{\mathrm{r}})$ as the challenge tuple.
- If $w' \in \Phi$, $\mathcal{A}_1''$ returns $(\Phi, \tau_{\mathtt{p}}, w_0 = w, w_1 = w')$ as the challenge tuple.

– $\mathcal{A}_2''$ answers $\mathcal{A}_3$'s secret functional key generation queries, and then returns $\mathcal{A}_3$'s final output as its own final output.

We analyze $\mathcal{A}''$'s advantage as follows.

First of all, the probability that $i^*_{\mathrm{gs}} = i^* \notin U_{\mathtt{cor}}$ is at least $\frac{1}{n}$. So the probability that $\mathcal{A}''$ does not abort is at least $\frac{1}{n}$.

In the analysis below, assume that $\mathcal{A}''$ does not abort (i.e., $i^*_{\mathrm{gs}} = i^* \notin U_{\mathtt{cor}}$).

If the challenger (in game $\mathsf{Exp}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}''}(\lambda)$) encrypts $w_0 = w$, it is obvious that $\mathcal{A}''$ perfectly simulates $\mathbf{G}_1$ for $\mathcal{A}$.

On the other hand, if the challenger (in game $\mathsf{Exp}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}''}(\lambda)$) encrypts w_1, when $w' \in \Phi$, $\mathcal{A}''$ perfectly simulates $\mathbf{G}_2$ for $\mathcal{A}$. For the case of $w' \notin \Phi$, we have the following claim with a postponed proof.

Claim. If the challenger (in game $\mathsf{Exp}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}''}(\lambda)$) encrypts w_1, when $w' \notin \Phi$, $\mathcal{A}''$ perfectly simulates $\mathbf{G}_2$ for $\mathcal{A}$.

Hence, when $i^*_{\mathrm{gs}} = i^* \notin U_{\mathtt{cor}}$, $\mathcal{A}''$ succeeds in distinguishing the SME ciphertext of w_0 and the SME ciphertext of w_1, if and only if $\mathcal{A}$ succeeds in distinguishing $\mathbf{G}_1$ and $\mathbf{G}_2$.

Thus, we derive that

$$\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}''}(\lambda) \geq \frac{1}{2n} |\Pr[\mathbf{G}_2 = 1] - \Pr[\mathbf{G}_1 = 1]|,$$

concluding the proof of Lemma 1

So what remains is to prove the above claim.

Specifically, recall that for all $f \in \mathbb{F}$, $f(\hat{w}) \in \{0,1\}$ if $\hat{w} \in \Phi$, and $f(\hat{w}) = \perp$ otherwise. For the challenge tuple $(i^* = i^*_{\mathrm{gs}}, \tau_{\mathtt{p}}, x = (\mathsf{com}, \Phi), w_x = (w, r_{\mathrm{com}}), w'_x = (w', r'_{\mathrm{com}}))$ output by $\mathcal{A}_2$, since $(x, w_x) \in \mathcal{R}_{\mathrm{sm}}$, we have that for all $f \in \mathbb{F}$, $f(w) \in \{0,1\}$. But on the other hand, if $w' \notin \Phi$, $f(w') = \perp \neq f(w)$ for all $f \in \mathbb{F}$. In this case, the restrictions on $\mathcal{A}$ in this game require that

(i) for each secret functional key generation query $(i', f', \tau'_{\mathtt{f}})$ made by $\mathcal{A}_2$, if $i' = i^*_{\mathrm{gs}}$, then $\mathsf{P}(\tau_{\mathtt{p}}, \tau'_{\mathtt{f}}) = 0$;
(ii) for each secret functional key generation query $(i', f', \tau'_{\mathtt{f}})$ made by $\mathcal{A}_3$, if $(i' = i^*_{\mathrm{gs}}) \wedge (\mathsf{P}(\tau_{\mathtt{p}}, \tau'_{\mathtt{f}}) = 1)$, the oracle $\mathcal{O}^{\mathsf{FKGen}}$ will return $\perp$; otherwise, it will return $sk_{i',f',\tau'_{\mathtt{f}}} \leftarrow \mathsf{SME.KGen}(pk_{i'}, sk_{i'}, S_{f'}, \tau'_{\mathtt{f}})$.

We turn to $\mathcal{A}''$'s simulation when the challenger (in game $\mathsf{Exp}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}''}(\lambda)$) encrypts w_1 and $w' \notin \Phi$. Specifically, we aim to demonstrate that (1) the challenge tuple generated by $\mathcal{A}_1''$ satisfies the requirements of $\mathsf{Exp}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}''}(\lambda)$, and (2) $\mathcal{A}_2''$ perfectly simulates the secret functional key generation oracle for $\mathcal{A}_3$.

Note that when $w' \notin \Phi$, $\mathcal{A}_1''$ returns $(\Phi, \tau_{\mathtt{p}}, w_0 = w, w_1 = w_{\mathrm{r}})$ as the challenge tuple (in $\mathsf{Exp}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}''}(\lambda)$), where $w_{\mathrm{r}} \leftarrow \Phi$. We also notice that $\mathcal{A}_1''$ makes a secret

key generation query $(S_{f'}, \tau'_{\mathtt{f}})$ (in $\mathsf{Exp}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}''}(\lambda)$), only if $\mathcal{A}_2$ makes a secret functional key generation query $(i^*_{\mathrm{gs}}, f', \tau'_{\mathtt{f}})$. According to the above restriction (i) on $\mathcal{A}$, we derive that each secret key generation query $(S_{f'}, \tau'_{\mathtt{f}})$ made by $\mathcal{A}''_1$ satisfies $\mathsf{P}(\tau_{\mathtt{p}}, \tau'_{\mathtt{f}}) = 0$. In other words, all of $\mathcal{A}''_1$'s secret key generation queries put no restriction on w_1[8], i.e., w_1 can be a uniformly sampled w_{r}. So the challenge tuple generated by $\mathcal{A}''_1$ satisfies the requirements of $\mathsf{Exp}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}''}(\lambda)$.

On the other hand, for each secret functional key generation query $(i', f', \tau'_{\mathtt{f}})$ made by $\mathcal{A}_3$,

- if $(i' = i^*_{\mathrm{gs}}) \wedge (\mathsf{P}(\tau_{\mathtt{p}}, \tau'_{\mathtt{f}}) = 1)$, according to the above restriction (ii) on $\mathcal{A}$, $\mathcal{A}''_2$ can return $\perp$ to $\mathcal{A}_3$ directly as a response;
- if $i' \neq i^*_{\mathrm{gs}}$, $\mathcal{A}''_2$ can answer this query by itself, since the $n-1$ secret keys $(sk_i)_{i \in [n] \setminus \{i^*_{\mathrm{gs}}\}}$ are all generated by $\mathcal{A}''$;
- if $(i' = i^*_{\mathrm{gs}}) \wedge (\mathsf{P}(\tau_{\mathtt{p}}, \tau'_{\mathtt{f}}) = 0)$, $\mathcal{A}''_2$ sends $(S_{f'}, \tau'_{\mathtt{f}})$ to its own secret key generation oracle (in game $\mathsf{Exp}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}''}(\lambda)$), and then return the response to $\mathcal{A}_3$.

Note that $\mathcal{A}''_2$ makes a secret key generation query $(S_{f'}, \tau'_{\mathtt{f}})$ (in $\mathsf{Exp}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}''}(\lambda)$), only if $\mathcal{A}_3$ makes a secret functional key generation query $(i^*_{\mathrm{gs}}, f', \tau'_{\mathtt{f}})$ satisfying $\mathsf{P}(\tau_{\mathtt{p}}, \tau'_{\mathtt{f}}) = 0$. In other words, each secret key generation query $(S_{f'}, \tau'_{\mathtt{f}})$ made by $\mathcal{A}''_2$ satisfies $\mathsf{P}(\tau_{\mathtt{p}}, \tau'_{\mathtt{f}}) = 0$. So the secret key generation oracle in (in $\mathsf{Exp}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}''}(\lambda)$) will not return $\perp$ as a response for these queries. Thus, $\mathcal{A}''_2$ perfectly simulates the secret functional key generation oracle for $\mathcal{A}_3$.

Hence, $\mathcal{A}''$ perfectly simulates $\mathbf{G}_2$ for $\mathcal{A}$, concluding the proof of the claim.

□

□

H Proof of Theorem 4

Proof (of Theorem 4). For any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ attacking IND security of SME, we construct a PPT adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ attacking adaptive security property of DIPE as follows.

Firstly, the challenger generates $(pk, msk) \leftarrow \mathsf{DIPE.Setup}(1^\lambda, (l_1, l_2)))$, samples $b \leftarrow \{0,1\}$, initializes $C_{\mathrm{y,m}} := \emptyset$ and $U_{\mathrm{x}} := \emptyset$, and sends $(pk, (l_1, l_2))$ to $\mathcal{B}_1$.

On input $(pk, (l_1, l_2))$, $\mathcal{B}_1$ initializes $W := \emptyset$ and $Q := \emptyset$, sends pk to $\mathcal{A}_1$, and then answers $\mathcal{A}_1$'s secret key generation query as below: for each query $(S' \subset [l], \tau'_{\mathtt{f}})$, $\mathcal{B}_1$ firstly sets $Q := Q \cup \{(S', \tau'_{\mathtt{f}})\}$, computes $I^*(S') \leftarrow \mathsf{EncodeS}(l, S')$, obtains $sk_{(I^*(S'), \tau'_{\mathtt{f}})}$ via querying its own secret key generation oracle (for dual IPE) on $(I^*(S'), \tau'_{\mathtt{f}})$, and then returns $sk_{S', \tau'_{\mathtt{f}}} := sk_{I^*(S'), \tau'_{\mathtt{f}}}$ to $\mathcal{A}_1$ as a response.

Note that for each $(S', \tau'_{\mathtt{f}}) \in Q$, the corresponding $(I^*(S'), \tau'_{\mathtt{f}})$ is added to U_{x} by the challenger.

[8] Note that if $\mathsf{P}(\tau_{\mathtt{p}}, \tau'_{\mathtt{f}}) = 1$ for some query $(S_{f'}, \tau'_{\mathtt{f}})$, then the challenge tuple $(\Phi, \tau_{\mathtt{p}}, w_0, w_1)$ returned by $\mathcal{A}''_1$ should satisfy that $\mathsf{func}_{\Phi_{S_{f'}}}(w_0) = \mathsf{func}_{\Phi_{S_{f'}}}(w_1)$.

Upon receiving $(\Phi, \tau_{\mathtt{p}}, w_0, w_1)$ from $\mathcal{A}_1$, $\mathcal{B}_1$ sets $W := \{\tau_{\mathtt{p}}, w_0, w_1\}$, computes $I(w_0) = \mathsf{EncodeW}(\Phi, w_0)$ and $I(w_1) = \mathsf{EncodeW}(\Phi, w_1)$, and then sends $(I(w_0), I(w_1), \tau_{\mathtt{p}}, \mathsf{m}_{\mathtt{dum}}, \mathsf{m}_{\mathtt{dum}})$ to the challenger.

Subsequently, the challenger sets $C_{\mathrm{y,m}} := \{(I(w_0), I(w_1), \tau_{\mathtt{p}}, \mathsf{m}_{\mathtt{dum}}, \mathsf{m}_{\mathtt{dum}})\}$, computes $c \leftarrow \mathsf{DIPE.Enc}(pk, (I(w_b), \tau_{\mathtt{p}}), \mathsf{m}_{\mathtt{dum}})$, and returns c to $\mathcal{B}_2$.

$\mathcal{B}_2$ sends c to $\mathcal{A}_2$, and then answers $\mathcal{A}_2$'s secret key generation queries as follows: for each query $(S' \subset [l], \tau'_{\mathtt{f}})$, if $(\mathsf{P}_{\mathrm{ip}}(\tau_{\mathtt{p}}, \tau'_{\mathtt{f}}) = 1) \wedge (\mathsf{func}_{\Phi_{S'}}(w_0) \neq \mathsf{func}_{\Phi_{S'}}(w_1))$, $\mathcal{B}_2$ returns $\perp$ to $\mathcal{A}_2$ as a response directly; otherwise, $\mathcal{B}_2$ sets $Q := Q \cup \{(S', \tau'_{\mathtt{f}})\}$, computes $I^*(S') \leftarrow \mathsf{EncodeS}(l, S')$, obtains $sk_{I^*(S'),\tau'_{\mathtt{f}}}$ via querying its own secret key generation oracle (for dual IPE) on $(I^*(S'), \tau'_{\mathtt{f}})$, and then returns $sk_{S',\tau'_{\mathtt{f}}} := sk_{I^*(S'),\tau'_{\mathtt{f}}}$ to $\mathcal{A}_2$ as a response.

Finally, receiving b' from $\mathcal{A}_2$, $\mathcal{B}_2$ returns b' as its own final output.

That is the construction of $\mathcal{B}$.

Now, we turn to analyze $\mathcal{B}$'s advantage.

We present the following two lemmas with postponed proofs.

Lemma 2. *The tuple* $(I(w_0), I(w_1), \tau_{\mathtt{p}}, \mathsf{m}_{\mathtt{dum}}, \mathsf{m}_{\mathtt{dum}})$*, which* $\mathcal{B}_1$ *sends to the challenger, satisfies that*

$$\begin{aligned}&(\forall \beta \in \{0,1\} : \mathtt{DuIP}((I^*(S'), \tau'_{\mathtt{f}}), (I(w_\beta), \tau_{\mathtt{p}})) \neq 0)\\ &\quad \vee (\forall \beta \in \{0,1\} : \mathtt{DuIP}((I^*(S'), \tau'_{\mathtt{f}}), (I(w_\beta), \tau_{\mathtt{p}})) = 0)\end{aligned}$$

for all $(I^*(S'), \tau'_{\mathtt{f}}) \in U_{\mathrm{x}}$.

Lemma 3. $\mathcal{B}$ *perfectly simulates the secret key generation oracle for* $\mathcal{A}$.

Combining these two lemmas, $\mathcal{B}$ perfectly simulates game $\mathsf{Exp}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}}(\lambda)$ for $\mathcal{A}$, and $\mathcal{B}$ wins game $\mathsf{Exp}^{\mathrm{as}}_{\mathsf{DIPE},\mathcal{B}}(\lambda)$ if and only if $\mathcal{A}$ wins game $\mathsf{Exp}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}}(\lambda)$. Hence, we obtain that

$$\mathbf{Adv}^{\mathrm{as}}_{\mathsf{DIPE},\mathcal{B}}(\lambda) = \mathbf{Adv}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}}(\lambda).$$

So, what remains is to prove the above two lemmas.

Proof (of Lemma 2). First of all, we stress that for the challenge tuple $(I(w_0), I(w_1), \tau_{\mathtt{p}}, \mathsf{m}_{\mathtt{dum}}, \mathsf{m}_{\mathtt{dum}})$ generated by $\mathcal{B}_1$, the challenge messages satisfy $m_0 = m_1 = \mathsf{m}_{\mathtt{dum}}$.

Note that $(S', \tau'_{\mathtt{f}}) \in Q$ if and only if $(I^*(S'), \tau'_{\mathtt{f}}) \in U_{\mathrm{x}}$. In game $\mathsf{Exp}^{\mathrm{ind}}_{\mathsf{SME},\mathcal{A}}(\lambda)$, the challenge tuple $(\Phi, \tau_{\mathtt{p}}, w_0, w_1)$ output by $\mathcal{A}_1$ is required to satisfy that (i) $(w_0 \in \Phi) \wedge (w_1 \in \Phi)$, and (ii) for all $(S', \tau'_{\mathtt{f}}) \in Q$ satisfying $\mathsf{P}_{\mathrm{ip}}(\tau_{\mathtt{p}}, \tau'_{\mathtt{f}}) = 1$, $\mathsf{func}_{\Phi_{S'}}(w_0) = \mathsf{func}_{\Phi_{S'}}(w_1)$. Note that $\mathsf{func}_{\Phi_{S'}}(w_0) = \mathsf{func}_{\Phi_{S'}}(w_1)$ guarantees that "$w_0 \in \Phi_{S'}$ if and only if $w_1 \in \Phi_{S'}$".

Hence, for any $((I^*(S'), \tau'_{\mathtt{f}}) \in U_{\mathrm{x}}$, there is $(S', \tau'_{\mathtt{f}}) \in Q$, and

- if $\mathsf{P}_{\mathrm{ip}}(\tau_{\mathtt{p}}, \tau'_{\mathtt{f}}) = 1$ (i.e., $\langle \tau_{\mathtt{p}}, \tau'_{\mathtt{f}} \rangle = 0$):
  - When $w_0 \in \Phi_{S'}$, we have $w_1 \in \Phi_{S'}$. In this case, according to the properties of $\mathsf{EncodeW}$ and $\mathsf{EncodeS}$, we obtain

$$\langle I^*(S'), I(w_0) \rangle = \langle I^*(S'), I(w_1) \rangle = 0.$$

  Thus, we derive that for all $\beta \in \{0,1\}$, $\mathtt{DuIP}((I^*(S'), \tau'_{\mathtt{f}}), (I(w_\beta), \tau_{\mathtt{p}})) = 0$.

- When $w_0 \notin \Phi_{S'}$, we have $w_1 \notin \Phi_{S'}$ either. In this case, according to the properties of $\mathsf{EncodeW}$ and $\mathsf{EncodeS}$, we obtain

$$\langle I^*(S'), I(w_0)\rangle = \langle I^*(S'), I(w_1)\rangle = 1.$$

Thus, we derive that for all $\beta \in \{0,1\}$, $\mathtt{DuIP}((I^*(S'), \tau'_{\mathtt{f}}), (I(w_\beta), \tau_{\mathtt{p}})) \neq 0$.

- if $\mathsf{P}_{\mathrm{ip}}(\tau_{\mathtt{p}}, \tau'_{\mathtt{f}}) = 0$ (i.e., $\langle \tau_{\mathtt{p}}, \tau'_{\mathtt{f}}\rangle \neq 0$): in this case, obviously for all $\beta \in \{0,1\}$, $\mathtt{DuIP}((I^*(S'), \tau'_{\mathtt{f}}), (I(w_\beta), \tau_{\mathtt{p}})) \neq 0$.

□

Proof (of Lemma 3). Obviously, $\mathcal{B}_1$ perfectly simulates the secret key generation oracle for $\mathcal{A}_1$. So this proof focuses on $\mathcal{B}_2$'s simulation.

For each of $\mathcal{A}_2$'s secret key generation query $(S', \tau'_{\mathtt{f}})$, if $(\mathsf{P}_{\mathrm{ip}}(\tau_{\mathtt{p}}, \tau'_{\mathtt{f}}) = 1) \wedge (\mathsf{func}_{\Phi_{S'}}(w_0) \neq \mathsf{func}_{\Phi_{S'}}(w_1))$, $\mathcal{B}_2$ returns $\perp$, perfectly simulating the oracle for $\mathcal{A}_2$; on the other hand,

- If $\mathsf{P}_{\mathrm{ip}}(\tau_{\mathtt{p}}, \tau'_{\mathtt{f}}) = 0$, which means $\langle \tau_{\mathtt{p}}, \tau'_{\mathtt{f}}\rangle \neq 0$, then for all $\beta \in \{0,1\}$,

$$(\mathtt{DuIP}((I^*(S'), \tau'_{\mathtt{f}}), (I(w_\beta), \tau_{\mathtt{p}})) \neq 0) \wedge (\mathtt{DuIP}((I^*(S'), \tau'_{\mathtt{f}}), (I(w_{1-\beta}), \tau_{\mathtt{p}})) \neq 0).$$

- If $\mathsf{func}_{\Phi_{S'}}(w_0) = \mathsf{func}_{\Phi_{S'}}(w_1)$, which means that "$w_0 \in \Phi_{S'}$ if and only if $w_1 \in \Phi_{S'}$", then $\langle I^*(S'), I(w_0)\rangle = \langle I^*(S'), I(w_1)\rangle$. Thus, we derive

$$\mathtt{DuIP}((I^*(S'), \tau'_{\mathtt{f}}), (I(w_0), \tau_{\mathtt{p}})) = \mathtt{DuIP}((I^*(S'), \tau'_{\mathtt{f}}), (I(w_1), \tau_{\mathtt{p}})).$$

Note that the challenge messages generated by $\mathcal{B}_1$ satisfy $m_0 = m_1 = \mathsf{m}_{\mathtt{dum}}$. So if $(\mathsf{P}_{\mathrm{ip}}(\tau_{\mathtt{p}}, \tau'_{\mathtt{f}}) = 0) \vee (\mathsf{func}_{\Phi_{S'}}(w_0) = \mathsf{func}_{\Phi_{S'}}(w_1))$, when $\mathcal{B}_2$ queries its own secret key generation oracle (for dual IPE) on $(I^*(S'), \tau'_{\mathtt{f}})$, it will receive $sk_{I^*(S'),\tau'_{\mathtt{f}}} \leftarrow \mathsf{KGen}(msk, (I^*(S'), \tau'_{\mathtt{f}}))$ (rather than $\perp$) as a response.

Hence, $\mathcal{B}_2$ also perfectly simulates the secret key generation oracle for $\mathcal{A}_2$. □

□

I Proof of Theorem 5

Proof (of Theorem 5). We will prove the theorem using a hybrid argument over a sequence of games. Following [12,32,10] without loss of generality, we assume that the messages m_0, m_1 submitted by the adversary at the challenge phase are equal, i.e., $m_0 = m_1 = m$, since we can reduce the case $m_0 \neq m_1$ to the case $m_0 = m_1$ by arguing that an encryption for m_0 is indistinguishable with an encryption for m_1. We also assume that the adversary makes at most q key queries.

Game_0 is the real game in which the challenge ciphertext for $\mathbf{y}^{(b)} = (\mathbf{y}_1^{(b)} = (y_{1,1}^{(b)}, \ldots, y_{1,l_1}^{(b)}), \mathbf{y}_2 = (y_{2,1}, \ldots, y_{2,l_2}))$ is of the form

$$(y_{1,1}^{(b)} \cdot \mathbf{u}_1 + \mathbf{w}_{1,1}, \ldots,\ y_{1,l_1}^{(b)} \cdot \mathbf{u}_1 + \mathbf{w}_{1,l_1},\ y_{2,1} \cdot \mathbf{u}_2 + \mathbf{w}_{2,1}, \ldots,\ y_{2,l_2} \cdot \mathbf{u}_2 + \mathbf{w}_{2,l_2},\ [\alpha]_2 \cdot m)$$

where $b \leftarrow \{0,1\}$.

$\mathbf{Game}_1$ is identical to $\mathbf{Game}_0$ except that the challenge ciphertext is $(y_{1,1}^{(b)} \cdot \mathbf{u}_1^{(13)} + y_{1,1}^{(1-b)} \cdot \mathbf{u}_1^{(2)} + \mathbf{w}_{1,1}, \ldots,\ y_{1,l_1}^{(b)} \cdot \mathbf{u}_1^{(13)} + y_{1,l_1}^{(1-b)} \cdot \mathbf{u}_1^{(2)} + \mathbf{w}_{1,l_1},\ y_{2,1} \cdot \mathbf{u}_2^{(13)} + y_{2,1} \cdot \mathbf{u}_2^{(2)} + \mathbf{w}_{2,1}, \ldots,\ y_{2,l_2} \cdot \mathbf{u}_2^{(13)} + y_{2,l_2} \cdot \mathbf{u}_2^{(2)} + \mathbf{w}_{2,l_2},\ [\alpha]_2 \cdot m)$.

$\mathbf{Game}_{2.j}$ for $j \in [0, q]$ is identical to $\mathbf{Game}_1$ except that the first j secret keys are of the form

$$([\alpha + (x_{1,1} \cdot \mathbf{w}_{1,1} + \cdots + x_{1,l_1} \cdot \mathbf{w}_{1,l_1} + x_{2,1} \cdot \mathbf{w}_{2,1} + \cdots + x_{2,l_2} \cdot \mathbf{w}_{2,l_2})\mathbf{d}]_2,\ [\mathbf{d}]_2)$$

where $\mathbf{d} \leftarrow \mathsf{span}_c(\mathbf{B}_1, \mathbf{B}_2)$.

$\mathbf{Game}_{2.j.1}$ for $j \in [0, q-1]$ is identical to $\mathbf{Game}_{2.j}$ except that the $(j+1)^{th}$ secret key is

$$([\alpha + (x_{1,1} \cdot \mathbf{w}_{1,1} + \cdots + x_{1,l_1} \cdot \mathbf{w}_{1,l_1} + x_{2,1} \cdot \mathbf{w}_{2,1} + \cdots + x_{2,l_2} \cdot \mathbf{w}_{2,l_2})\mathbf{d}]_2,\ [\mathbf{d}]_2)$$

where $\mathbf{d} \leftarrow \mathsf{span}_c(\mathbf{B}_1, \mathbf{B}_3)$.

$\mathbf{Game}_{2.j.2}$ is identical to $\mathbf{Game}_{2.j.1}$ except that the challenge ciphertext is $(y_{1,1}^{(b)} \cdot \mathbf{u}_1^{(1)} + y_{1,1}^{(1-b)} \cdot \mathbf{u}_1^{(2)} + y_{1,1}^{(1-b)} \cdot \mathbf{u}_1^{(3)} + \mathbf{w}_{1,1}, \ldots,\ y_{1,l_1}^{(b)} \cdot \mathbf{u}_1^{(1)} + y_{1,l_1}^{(1-b)} \cdot \mathbf{u}_1^{(2)} + y_{1,l_1}^{(1-b)} \cdot \mathbf{u}_1^{(3)} + \mathbf{w}_{1,l_1},\ y_{2,1} \cdot \mathbf{u}_2^{(1)} + y_{2,1} \cdot \mathbf{u}_2^{(2)} + y_{2,1} \cdot \mathbf{u}_2^{(3)} + \mathbf{w}_{2,1}, \ldots,\ y_{2,l_2} \cdot \mathbf{u}_2^{(1)} + y_{2,l_2} \cdot \mathbf{u}_2^{(2)} + y_{2,l_2} \cdot \mathbf{u}_2^{(3)} + \mathbf{w}_{2,l_2},\ [\alpha]_2 \cdot m)$.

$\mathbf{Game}_{2.j.3}$ for $j \in [0, q-1]$ is identical to $\mathbf{Game}_{2.j.2}$ except that the $(j+1)^{th}$ secret key is

$$([\alpha + (x_{1,1} \cdot \mathbf{w}_{1,1} + \cdots + x_{1,l_1} \cdot \mathbf{w}_{1,l_1} + x_{2,1} \cdot \mathbf{w}_{2,1} + \cdots + x_{2,l_2} \cdot \mathbf{w}_{2,l_2})\mathbf{d}]_2,\ [\mathbf{d}]_2)$$

where $\mathbf{d} \leftarrow \mathsf{span}_c(\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3)$.

$\mathbf{Game}_{2.j.4}$ is identical to $\mathbf{Game}_{2.j.3}$ except that the challenge ciphertext is $(y_{1,1}^{(b)} \cdot \mathbf{u}_1^{(1)} + y_{1,1}^{(1-b)} \cdot \mathbf{u}_1^{(2)} + y_{1,1}^{(b)} \cdot \mathbf{u}_1^{(3)} + \mathbf{w}_{1,1}, \ldots,\ y_{1,l_1}^{(b)} \cdot \mathbf{u}_1^{(1)} + y_{1,l_1}^{(1-b)} \cdot \mathbf{u}_1^{(2)} + y_{1,l_1}^{(b)} \cdot \mathbf{u}_1^{(3)} + \mathbf{w}_{1,l_1},\ y_{2,1} \cdot \mathbf{u}_2^{(1)} + y_{2,1} \cdot \mathbf{u}_2^{(2)} + y_{2,1} \cdot \mathbf{u}_2^{(3)} + \mathbf{w}_{2,1}, \ldots,\ y_{2,l_2} \cdot \mathbf{u}_2^{(1)} + y_{2,l_2} \cdot \mathbf{u}_2^{(2)} + y_{2,l_2} \cdot \mathbf{u}_2^{(3)} + \mathbf{w}_{2,l_2},\ [\alpha]_2 \cdot m)$.

$\mathbf{Game}_{2.j.5}$ for $j \in [0, q-1]$ is identical to $\mathbf{Game}_{2.j.4}$ except that the $(j+1)^{th}$ secret key is

$$([\alpha + (x_{1,1} \cdot \mathbf{w}_{1,1} + \cdots + x_{1,l_1} \cdot \mathbf{w}_{1,l_1} + x_{2,1} \cdot \mathbf{w}_{2,1} + \cdots + x_{2,l_2} \cdot \mathbf{w}_{2,l_2})\mathbf{d}]_2,\ [\mathbf{d}]_2)$$

where $\mathbf{d} \leftarrow \mathsf{span}_c(\mathbf{B}_1, \mathbf{B}_2)$.

$\mathbf{Game}_3$ is identical to $\mathbf{Game}_{2.q}$ except that the challenge ciphertext is $(y_{1,1}^{(0)} \cdot \mathbf{u}_{1,0}^{(12)} + y_{1,1}^{(1)} \cdot \mathbf{u}_{1,1}^{(12)} + y_{1,1}^{(b)} \cdot \mathbf{u}_1^{(3)} + \mathbf{w}_{1,1}, \ldots,\ y_{1,l_1}^{(0)} \cdot \mathbf{u}_{1,0}^{(12)} + y_{1,l_1}^{(1)} \cdot \mathbf{u}_{1,1}^{(12)} + y_{1,l_1}^{(b)} \cdot \mathbf{u}_1^{(3)} + \mathbf{w}_{1,l_1},\ y_{2,1} \cdot \mathbf{u}_2^{(1)} + y_{2,1} \cdot \mathbf{u}_2^{(2)} + y_{2,1} \cdot \mathbf{u}_2^{(3)} + \mathbf{w}_{2,1}, \ldots,\ y_{2,l_2} \cdot \mathbf{u}_2^{(1)} + y_{2,l_2} \cdot \mathbf{u}_2^{(2)} + y_{2,l_2} \cdot \mathbf{u}_2^{(3)} + \mathbf{w}_{2,l_2},\ [\alpha]_2 \cdot m)$, where $\mathbf{u}_{1,0}, \mathbf{u}_{1,1} \leftarrow \mathbb{Z}_p^{1 \times (2k+1)}$.

$\mathbf{Game}_4$ is identical to $\mathbf{Game}_3$ except that the challenge ciphertext is $(y_{1,1}^{(0)} \cdot \mathbf{u}_{1,0} + y_{1,1}^{(1)} \cdot \mathbf{u}_{1,1} + \mathbf{w}_{1,1}, \ldots,\ y_{1,l_1}^{(0)} \cdot \mathbf{u}_{1,0} + y_{1,l_1}^{(1)} \cdot \mathbf{u}_{1,1} + \mathbf{w}_{1,l_1},\ y_{2,1} \cdot \mathbf{u}_2^{(1)} + y_{2,1} \cdot \mathbf{u}_2^{(2)} + y_{2,1} \cdot \mathbf{u}_2^{(3)} + \mathbf{w}_{2,1}, \ldots,\ y_{2,l_2} \cdot \mathbf{u}_2^{(1)} + y_{2,l_2} \cdot \mathbf{u}_2^{(2)} + y_{2,l_2} \cdot \mathbf{u}_2^{(3)} + \mathbf{w}_{2,l_2},\ [\alpha]_2 \cdot m)$.

We prove these games are indistinguishable in the following lemmas. Note that $\mathsf{Game}_{2.0} = \mathsf{Game}_1$ and $\mathsf{Game}_{2.j} = \mathsf{Game}_{2.j-1.5}$. In Game_4, it is clear that the value of b is information-theoretically hidden from the adversary. Hence the adversary has no advantage in Game_4. Therefore, we conclude that the advantage of the adversary in Game_0 is negligible. This completes the proof.

Lemma 4. Game_1 *is identical to* Game_0.

Proof. By the facts shown in Section 5.3, it is implied by the statement that, for all $\mathbf{u}_1^{(2)} \in \mathsf{span}_r(\mathbf{B}_2^{*\top})$, it holds that

$$\{y_{1,i}^{(b)} \cdot \mathbf{u}_1^{(2)} + \mathbf{w}_{1,i}^{(2)}\}_{i\in[l_1]} \equiv \{\mathbf{w}_{1,i}^{(2)}\}_{i\in[l_1]} \equiv \{y_{1,i}^{(1-b)} \cdot \mathbf{u}_1^{(2)} + \mathbf{w}_{1,i}^{(2)}\}_{i\in[l_1]},$$

when $\{\mathbf{w}_{1,i}^{(2)}\}_{i\in[l_1]} \leftarrow \mathsf{span}_r(\mathbf{B}_2^{*\top})$. Therefore, for all $\mathbf{u}_1 \leftarrow \mathbb{Z}_p^{1\times(2k+1)}$ it holds that

$$\begin{aligned}
&(\mathbf{w}_{1,1}\mathbf{B}_1, \dots, \mathbf{w}_{1,l_1}\mathbf{B}_1, \mathbf{w}_{2,1}\mathbf{B}_1, \dots, \mathbf{w}_{2,l_2}\mathbf{B}_1,\\
&y_{1,1}^{(b)} \cdot \mathbf{u}_1^{(13)} + y_{1,1}^{(b)} \cdot \mathbf{u}_1^{(2)} + \mathbf{w}_{1,1}, \dots,\ y_{1,l_1}^{(b)} \cdot \mathbf{u}_1^{(13)} + y_{1,l_1}^{(b)} \cdot \mathbf{u}_1^{(2)} + \mathbf{w}_{1,l_1},\\
&y_{2,1} \cdot \mathbf{u}_2^{(13)} + y_{2,1} \cdot \mathbf{u}_2^{(2)} + \mathbf{w}_{2,1}, \dots,\ y_{2,l_2} \cdot \mathbf{u}_2^{(13)} + y_{2,l_2} \cdot \mathbf{u}_2^{(2)} + \mathbf{w}_{2,l_2})\\
\equiv\ &(\mathbf{w}_{1,1}\mathbf{B}_1, \dots, \mathbf{w}_{1,l_1}\mathbf{B}_1, \mathbf{w}_{2,1}\mathbf{B}_1, \dots, \mathbf{w}_{2,l_2}\mathbf{B}_1,\\
&y_{1,1}^{(b)} \cdot \mathbf{u}_1^{(13)} + y_{1,1}^{(1-b)} \cdot \mathbf{u}_1^{(2)} + \mathbf{w}_{1,1}, \dots,\ y_{1,l_1}^{(b)} \cdot \mathbf{u}_1^{(13)} + y_{1,l_1}^{(1-b)} \cdot \mathbf{u}_1^{(2)} + \mathbf{w}_{1,l_1},\\
&y_{2,1} \cdot \mathbf{u}_2^{(13)} + y_{2,1} \cdot \mathbf{u}_2^{(2)} + \mathbf{w}_{2,1}, \dots,\ y_{2,l_2} \cdot \mathbf{u}_2^{(13)} + y_{2,l_2} \cdot \mathbf{u}_2^{(2)} + \mathbf{w}_{2,l_2})
\end{aligned}$$

This completes the proof. □

Lemma 5. *Under the* $\mathrm{SD}^{\mathbb{G}_2}_{\mathbf{B}_1\mapsto\mathbf{B}_1,\mathbf{B}_3}$ *assumption, for each* $j \in [q]$, $\mathsf{Game}_{2.j-1}$ *and* $\mathsf{Game}_{2.j-1.1}$ *are computationally indistinguishable.*

Proof. Suppose there exists an algorithm $\mathcal{A}$ that distinguishes $\mathsf{Game}_{2.j-1}$ and $\mathsf{Game}_{2.j-1.1}$. Then we can build an algorithm $\mathcal{B}$ breaking $\mathrm{SD}^{\mathbb{G}_2}_{\mathbf{B}_1\mapsto\mathbf{B}_1,\mathbf{B}_3}$ assumption with non-negligible advantage. $\mathcal{B}$ is given $[\mathbf{B}_1]_2$, $[\mathbf{B}_2]_2$, $[\mathbf{B}_3]_2$, $\mathsf{basis}(\mathbf{B}_2^*)$, $\mathsf{basis}(\mathbf{B}_1^*, \mathbf{B}_3^*)$, $[\mathbf{t}]_2$ and going to tell $\mathbf{t} \leftarrow \mathsf{span}_c(\mathbf{B}_1)$ or $\mathbf{t} \leftarrow \mathsf{span}_c(\mathbf{B}_1, \mathbf{B}_3)$. $\mathcal{B}$ will simulate $\mathsf{Game}_{2.j-1}$ or $\mathsf{Game}_{2.j-1.1}$ for $\mathcal{A}$. First $\mathcal{B}$ chooses $\alpha \leftarrow \mathbb{Z}_p$, $\mathbf{w}_{1,1}, \dots, \mathbf{w}_{1,l_1}, \mathbf{w}_{2,1}, \dots, \mathbf{w}_{2,l_2} \leftarrow \mathbb{Z}_p^{1\times(2k+1)}$, and picks $\mathbf{u}_1^{(13)}, \mathbf{u}_2^{(13)} \leftarrow \mathsf{span}_r((\mathbf{B}_1^*|\mathbf{B}_3^*)^\top)$ and $\mathbf{u}_1^{(2)}, \mathbf{u}_2^{(2)} \leftarrow \mathsf{span}_r(\mathbf{B}_2^{*\top})$ using $\mathsf{basis}(\mathbf{B}_1^*, \mathbf{B}_3^*)$ and $\mathsf{basis}(\mathbf{B}_2^*)$, respectively.

When $\mathcal{A}$ makes the κth key query, $\mathcal{B}$ uses $[\mathbf{B}_1]_2$, $[\mathbf{B}_2]_2$, $[\mathbf{t}]_2$ to output

$$([\alpha + (x_{1,1} \cdot \mathbf{w}_{1,1} + \cdots + x_{1,l_1} \cdot \mathbf{w}_{1,l_1} + x_{2,1} \cdot \mathbf{w}_{2,1} + \cdots + x_{2,l_2} \cdot \mathbf{w}_{2,l_2})\mathbf{d}]_2,\ [\mathbf{d}]_2),$$

where

$$\mathbf{d} \leftarrow \begin{cases} \mathsf{span}_c(\mathbf{B}_1, \mathbf{B}_2) & \kappa < j \\ \mathbf{t} & \kappa = j \\ \mathsf{span}_c(\mathbf{B}_1) & \kappa > j \end{cases}.$$

At some point, when $\mathcal{A}$ submits the challenge $(\mathbf{y}^{(0)} = (\mathbf{y}_1^{(0)}, \mathbf{y}_2), \mathbf{y}^{(1)} = (\mathbf{y}_1^{(1)}, \mathbf{y}_2), m_0, m_1)$ with $m_0 = m_1 = m$, $\mathcal{B}$ outputs $(y_{1,1}^{(b)} \cdot \mathbf{u}_1^{(13)} + y_{1,1}^{(1-b)} \cdot \mathbf{u}_1^{(2)} + \mathbf{w}_{1,1}, \dots,$

$y_{1,l_1}^{(b)} \cdot \mathbf{u}_1^{(13)} + y_{1,l_1}^{(1-b)} \cdot \mathbf{u}_1^{(2)} + \mathbf{w}_{1,l_1}$, $y_{2,1} \cdot \mathbf{u}_2^{(13)} + y_{2,1} \cdot \mathbf{u}_2^{(2)} + \mathbf{w}_{2,1}$, …, $y_{2,l_2} \cdot \mathbf{u}_2^{(13)} + y_{2,l_2} \cdot \mathbf{u}_2^{(2)} + \mathbf{w}_{2,l_2}), [\alpha]_2 \cdot m)$.

Observe that, when $\mathbf{t}$ is uniformly distributed over $\mathsf{span}_c(\mathbf{B}_1)$, $\mathcal{B}$ has properly simulated $\mathbf{Game}_{2.j-1}$; otherwise, when $\mathbf{t}$ is uniformly distributed over $\mathsf{span}_c(\mathbf{B}_1, \mathbf{B}_3)$, $\mathcal{B}$ has properly simulated $\mathbf{Game}_{2.j-1.1}$. This proves the lemma. □

Lemma 6. *For each $j \in [q]$, $\mathbf{Game}_{2.j-1.1}$ is identical to $\mathbf{Game}_{2.j-1.2}$.*

Proof. It is sufficient to prove the following statement: for all $\mathbf{y}^{(0)} = (\mathbf{y}_1^{(0)}, \mathbf{y}_2)$, $\mathbf{y}^{(1)} = (\mathbf{y}_1^{(1)}, \mathbf{y}_2)$, and $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ (corresponding to the jth key query) satisfying that: 1) $\mathtt{DuIP}((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{y}_1^{(0)}, \mathbf{y}_2)) = \mathtt{DuIP}((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{y}_1^{(1)}, \mathbf{y}_2)) = 0$; or 2) $\mathtt{DuIP}((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{y}_1^{(0)}, \mathbf{y}_2)) \neq 0 \wedge \mathtt{DuIP}((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{y}_1^{(1)}, \mathbf{y}_2)) \neq 0$, it holds that

$$
\begin{aligned}
&(y_{1,1}^{(b)} \cdot \mathbf{u}_1^{(3)} + \mathbf{w}_{1,1}^{(3)}, \ldots,\ y_{1,l_1}^{(b)} \cdot \mathbf{u}_1^{(3)} + \mathbf{w}_{1,l_1}^{(3)},\\
&\quad y_{2,1} \cdot \mathbf{u}_2^{(3)} + \mathbf{w}_{2,1}^{(3)}, \ldots,\ y_{2,l_2} \cdot \mathbf{u}_2^{(3)} + \mathbf{w}_{2,l_2}^{(3)},\\
&\quad x_{1,1} \cdot \mathbf{w}_{1,1}^{(3)} + \cdots + x_{1,l_1} \cdot \mathbf{w}_{1,l_1}^{(3)} + x_{2,1} \cdot \mathbf{w}_{2,1}^{(3)} + \cdots + x_{2,l_2} \cdot \mathbf{w}_{2,l_2}^{(3)})\\
\equiv\ &(y_{1,1}^{(1-b)} \cdot \mathbf{u}_1^{(3)} + \mathbf{w}_{1,1}^{(3)}, \ldots,\ y_{1,l_1}^{(1-b)} \cdot \mathbf{u}_1^{(3)} + \mathbf{w}_{1,l_1}^{(3)},\\
&\quad y_{2,1} \cdot \mathbf{u}_2^{(3)} + \mathbf{w}_{2,1}^{(3)}, \ldots,\ y_{2,l_2} \cdot \mathbf{u}_2^{(3)} + \mathbf{w}_{2,l_2}^{(3)},\\
&\quad x_{1,1} \cdot \mathbf{w}_{1,1}^{(3)} + \cdots + x_{1,l_1} \cdot \mathbf{w}_{1,l_1}^{(3)} + x_{2,1} \cdot \mathbf{w}_{2,1}^{(3)} + \cdots + x_{2,l_2} \cdot \mathbf{w}_{2,l_2}^{(3)})
\end{aligned}
$$

when $\mathbf{u}_1^{(3)}$, $\mathbf{u}_2^{(3)}$, $\mathbf{w}_{1,1}^{(3)}$, …, $\mathbf{w}_{1,l_1}^{(3)}$, $\mathbf{w}_{2,1}^{(3)}$, …, $\mathbf{w}_{2,l_2}^{(3)} \leftarrow \mathsf{span}_r(\mathbf{B}_3^{*\top})$. By the linearity, it in turn follows from the following statement

$$
\begin{aligned}
&\{y_{1,1}^{(b)} \cdot u_1 + w_{1,1}, \ldots,\ y_{1,l_1}^{(b)} \cdot u_1 + w_{1,l_1},\\
&\quad y_{2,1} \cdot u_2 + w_{2,1}, \ldots,\ y_{2,l_2} \cdot u_2 + w_{2,l_2},\\
&\quad x_{1,1} \cdot w_{1,1} + \cdots + x_{1,l_1} \cdot w_{1,l_1} + x_{2,1} \cdot w_{2,1} + \cdots + x_{2,l_2} \cdot w_{2,l_2}\}\\
\equiv\ &\{y_{1,1}^{(1-b)} \cdot u_1 + w_{1,1}, \ldots,\ y_{1,l_1}^{(1-b)} \cdot u_1 + w_{1,l_1},\\
&\quad y_{2,1} \cdot u_2 + w_{2,1}, \ldots,\ y_{2,l_2} \cdot u_2 + w_{2,l_2},\\
&\quad x_{1,1} \cdot w_{1,1} + \cdots + x_{1,l_1} \cdot w_{1,l_1} + x_{2,1} \cdot w_{2,1} + \cdots + x_{2,l_2} \cdot w_{2,l_2}\}
\end{aligned}
$$

when $u_1, u_2, w_{1,1}, \ldots, w_{1,l_1}, w_{2,1}, \ldots, w_{2,l_2} \leftarrow \mathbb{Z}_p$. This follows from the statistical argument for all $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$, by programming $\tilde{w}_{1,i} = y_{1,i} \cdot u_1 + w_{1,i}$ for all $i \in [l_1]$ and $\tilde{w}_{2,i} = y_{2,i} \cdot u_2 + w_{2,i}$ for all $i \in [l_2]$, we have

$$
\begin{aligned}
&\{y_{1,1} \cdot u_1 + w_{1,1}, \ldots,\ y_{1,l_1} \cdot u_1 + w_{1,l_1},\\
&\quad y_{2,1} \cdot u_2 + w_{2,1}, \ldots,\ y_{2,l_2} \cdot u_2 + w_{2,l_2},\\
&\quad x_{1,1} \cdot w_{1,1} + \cdots + x_{1,l_1} \cdot w_{1,l_1} + x_{2,1} \cdot w_{2,1} + \cdots + x_{2,l_2} \cdot w_{2,l_2}\}\\
\equiv\ &\{\tilde{w}_{1,1}, \ldots, \tilde{w}_{1,l_1},\\
&\quad \tilde{w}_{2,1}, \ldots, \tilde{w}_{2,l_2},\\
&\quad (x_{1,1} \cdot \tilde{w}_{1,1} + \cdots + x_{1,l_1} \cdot \tilde{w}_{1,l_1} + x_{2,1} \cdot \tilde{w}_{2,1} + \cdots + x_{2,l_2} \cdot \tilde{w}_{2,l_2}) - u_1 \cdot \langle \mathbf{x}_1, \mathbf{y}_1 \rangle - u_2 \cdot \langle \mathbf{x}_2, \mathbf{y}_2 \rangle\}
\end{aligned}
$$

which means that the left-hand side distributions for all vector $\mathbf{y}$ are identical if $\mathtt{DuIP}((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{y}_1, \mathbf{y}_2)) \neq 0$(since u_1, u_2 hide the information about the inner-product), and so do all vector $\mathbf{y}$ if $\mathtt{DuIP}((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{y}_1, \mathbf{y}_2)) = 0$. This immediately proves the above statement and thus proves the lemma. □

Lemma 7. *Under the* $\mathrm{SD}^{\mathbb{G}_2}_{\mathbf{B}_3 \mapsto \mathbf{B}_2, \mathbf{B}_3}$ *assumption, for each* $j \in [q]$, $\mathbf{Game}_{2.j-1.2}$ *and* $\mathbf{Game}_{2.j-1.3}$ *are computationally indistinguishable.*

Proof. Suppose there exists an algorithm $\mathcal{A}$ that distinguishes $\mathbf{Game}_{2.j-1.2}$ and $\mathbf{Game}_{2.j-1.3}$. Then we can build an algorithm $\mathcal{B}$ breaking $\mathrm{SD}^{\mathbb{G}_2}_{\mathbf{B}_3 \mapsto \mathbf{B}_2, \mathbf{B}_3}$ assumption with non-negligible advantage. $\mathcal{B}$ is given $[\mathbf{B}_1]_2$, $[\mathbf{B}_2]_2$, $[\mathbf{B}_3]_2$, $\mathsf{basis}(\mathbf{B}_1^*)$, $\mathsf{basis}(\mathbf{B}_2^*, \mathbf{B}_3^*)$, $[\mathbf{t}]_2$ and going to tell $\mathbf{t} \leftarrow \mathsf{span}(\mathbf{B}_3)$ or $\mathbf{t} \leftarrow \mathsf{span}(\mathbf{B}_2, \mathbf{B}_3)$. $\mathcal{B}$ will simulate $\mathbf{Game}_{2.j-1.2}$ or $\mathbf{Game}_{2.j-1.3}$ for $\mathcal{A}$. First $\mathcal{B}$ chooses $\alpha \leftarrow \mathbb{Z}_p$, $\mathbf{w}_{1,1}, \ldots, \mathbf{w}_{1,l_1}, \mathbf{w}_{2,1}, \ldots, \mathbf{w}_{2,l_2} \leftarrow \mathbb{Z}_p^{1 \times (2k+1)}$, and picks $\mathbf{u}_1^{(1)}, \mathbf{u}_2^{(1)} \leftarrow \mathsf{span}_r(\mathbf{B}_1^{*\top})$ and $\mathbf{u}_1^{(23)}, \mathbf{u}_2^{(23)} \leftarrow \mathsf{span}_r((\mathbf{B}_2^* | \mathbf{B}_3^*)^\top)$ using $\mathsf{basis}(\mathbf{B}_1^*)$ and $\mathsf{basis}(\mathbf{B}_2^*, \mathbf{B}_3^*)$, respectively.

When $\mathcal{A}$ makes the κth key query, $\mathcal{B}$ uses $[\mathbf{B}_1]_2$, $[\mathbf{B}_2]_2$, $[\mathbf{t}]_2$ to output

$$([\alpha + (x_{1,1} \cdot \mathbf{w}_{1,1} + \cdots + x_{1,l_1} \cdot \mathbf{w}_{1,l_1} + x_{2,1} \cdot \mathbf{w}_{2,1} + \cdots + x_{2,l_2} \cdot \mathbf{w}_{2,l_2})\mathbf{d}]_2,\ [\mathbf{d}]_2)$$

where

$$\mathbf{d} \leftarrow \begin{cases} \mathsf{span}_c(\mathbf{B}_1, \mathbf{B}_2) & \kappa < j \\ \mathbf{t} + \mathsf{span}_c(\mathbf{B}_1) & \kappa = j \\ \mathsf{span}_c(\mathbf{B}_1) & \kappa > j \end{cases}.$$

At some point, when $\mathcal{A}$ submits the challenge $(\mathbf{y}^{(0)} = (\mathbf{y}_1^{(0)}, \mathbf{y}_2), \mathbf{y}^{(1)} = (\mathbf{y}_1^{(1)}, \mathbf{y}_2), m_0, m_1)$ with $m_0 = m_1 = m$, $\mathcal{B}$ outputs $(y_{1,1}^{(b)} \cdot \mathbf{u}_1^{(1)} + y_{1,1}^{(1-b)} \cdot \mathbf{u}_1^{(23)} + \mathbf{w}_{1,1}, \ldots, y_{1,l_1}^{(b)} \cdot \mathbf{u}_1^{(1)} + y_{1,l_1}^{(1-b)} \cdot \mathbf{u}_1^{(23)} + \mathbf{w}_{1,l_1}, y_{2,1} \cdot \mathbf{u}_2^{(1)} + y_{2,1} \cdot \mathbf{u}_2^{(23)} + \mathbf{w}_{2,1}, \ldots, y_{2,l_2} \cdot \mathbf{u}_2^{(1)} + y_{2,l_2} \cdot \mathbf{u}_2^{(23)} + \mathbf{w}_{2,l_2}), [\alpha]_2 \cdot m)$.

Observe that, when $\mathbf{t}$ is uniformly distributed over $\mathsf{span}_c(\mathbf{B}_3)$, $\mathcal{B}$ has properly simulated $\mathbf{Game}_{2.j-1.2}$; otherwise, when $\mathbf{t}$ is uniformly distributed over $\mathsf{span}_c(\mathbf{B}_2, \mathbf{B}_3)$, $\mathcal{B}$ has properly simulated $\mathbf{Game}_{2.j-1.3}$. This proves the lemma. □

Lemma 8. *For each* $j \in [q]$, $\mathbf{Game}_{2.j-1.3}$ *is identical to* $\mathbf{Game}_{2.j-1.4}$.

Proof. The proof is identical to that for Lemma 6. □

Lemma 9. *Under the* $\mathrm{SD}^{\mathbb{G}_2}_{\mathbf{B}_1 \mapsto \mathbf{B}_1, \mathbf{B}_3}$ *assumption, for each* $j \in [q]$, $\mathbf{Game}_{2.j-1.4}$ *and* $\mathbf{Game}_{2.j-1.1.5}$ *are computationally indistinguishable.*

Proof. Suppose there exists an algorithm $\mathcal{A}$ that distinguishes $\mathbf{Game}_{2.j-1.4}$ and $\mathbf{Game}_{2.j-1.1.5}$. Then we can build an algorithm $\mathcal{B}$ breaking $\mathrm{SD}^{\mathbb{G}_2}_{\mathbf{B}_1 \mapsto \mathbf{B}_1, \mathbf{B}_3}$ assumption with non-negligible advantage. $\mathcal{B}$ is given $[\mathbf{B}_1]_2$, $[\mathbf{B}_2]_2$, $[\mathbf{B}_3]_2$, $\mathsf{basis}(\mathbf{B}_2^*)$, $\mathsf{basis}(\mathbf{B}_1^*, \mathbf{B}_3^*)$, $[\mathbf{t}]_2$ and going to tell $\mathbf{t} \leftarrow \mathsf{span}(\mathbf{B}_1)$ or $\mathbf{t} \leftarrow \mathsf{span}(\mathbf{B}_1, \mathbf{B}_3)$. $\mathcal{B}$ will simulate $\mathbf{Game}_{2.j-1.4}$ or $\mathbf{Game}_{2.j-1.1.5}$ for $\mathcal{A}$. First $\mathcal{B}$ chooses $\alpha \leftarrow \mathbb{Z}_p$,

$\mathbf{w}_{1,1},\ldots,\mathbf{w}_{1,l_1},\mathbf{w}_{2,1},\ldots,\mathbf{w}_{2,l_2} \leftarrow \mathbb{Z}_p^{1\times(2k+1)}$, and picks $\mathbf{u}_1^{(13)},\mathbf{u}_2^{(13)} \leftarrow \mathsf{span}_r((\mathbf{B}_1^*|\mathbf{B}_3^*)^\top)$ and $\mathbf{u}_1^{(2)},\mathbf{u}_2^{(2)} \leftarrow \mathsf{span}_r(\mathbf{B}_2^{*\top})$ using $\mathsf{basis}(\mathbf{B}_1^*,\mathbf{B}_3^*)$ and $\mathsf{basis}(\mathbf{B}_2^*)$, respectively.

When $\mathcal{A}$ makes the κth key query, $\mathcal{B}$ uses $[\mathbf{B}_1]_2$, $[\mathbf{B}_2]_2$, $[\mathbf{t}]_2$ to output

$$([\alpha+(x_{1,1}\cdot\mathbf{w}_{1,1}+\cdots+x_{1,l_1}\cdot\mathbf{w}_{1,l_1}+x_{2,1}\cdot\mathbf{w}_{2,1}+\cdots+x_{2,l_2}\cdot\mathbf{w}_{2,l_2})\mathbf{d}]_2,\ [\mathbf{d}]_2)$$

where

$$\mathbf{d}\leftarrow\begin{cases}\mathsf{span}_c(\mathbf{B}_1,\mathbf{B}_2) & \kappa<j\\ \mathbf{t}+\mathsf{span}_c(\mathbf{B}_2) & \kappa=j\\ \mathsf{span}_c(\mathbf{B}_1) & \kappa>j\end{cases}.$$

At some point, when $\mathcal{A}$ submits the challenge $(\mathbf{y}^{(0)}=(\mathbf{y}_1^{(0)},\mathbf{y}_2),\mathbf{y}^{(1)}=(\mathbf{y}_1^{(1)},\mathbf{y}_2),m_0,m_1)$ with $m_0=m_1=m$, $\mathcal{B}$ outputs $(y_{1,1}^{(b)}\cdot\mathbf{u}_1^{(13)}+y_{1,1}^{(1-b)}\cdot\mathbf{u}_1^{(2)}+\mathbf{w}_{1,1},\ \ldots,\ y_{1,l_1}^{(b)}\cdot\mathbf{u}_1^{(13)}+y_{1,l_1}^{(1-b)}\cdot\mathbf{u}_1^{(2)}+\mathbf{w}_{1,l_1},\ y_{2,1}\cdot\mathbf{u}_2^{(13)}+y_{2,1}\cdot\mathbf{u}_2^{(2)}+\mathbf{w}_{2,1},\ \ldots,\ y_{2,l_2}\cdot\mathbf{u}_2^{(13)}+y_{2,l_2}\cdot\mathbf{u}_2^{(2)}+\mathbf{w}_{2,l_2}),\ [\alpha]_2\cdot m)$.

Observe that, when $\mathbf{t}$ is uniformly distributed over $\mathsf{span}_c(\mathbf{B}_1)$, $\mathcal{B}$ has properly simulated $\mathbf{Game}_{2.j-1.5}$; otherwise, when $\mathbf{t}$ is uniformly distributed over $\mathsf{span}_c(\mathbf{B}_1,\mathbf{B}_3)$, $\mathcal{B}$ has properly simulated $\mathbf{Game}_{2.j-1.1.4}$. This proves the lemma. □

Lemma 10. $\mathbf{Game}_{2.q}$ *is identical to* $\mathbf{Game}_3$.

Proof. We choose $\widetilde{\mathbf{B}}_1,\mathbf{B}_3 \leftarrow \mathbb{Z}_p^{(2k+1)\times k}$, $\widetilde{\mathbf{B}}_2 \leftarrow \mathbb{Z}_p^{2k+1}$ and compute dual basis $\widetilde{\mathbf{B}}_1^*,\widetilde{\mathbf{B}}_2^*,\mathbf{B}_3^*$ as usual. Pick $\mathbf{R}\leftarrow \mathrm{GL}_{k+1}(\mathbb{Z}_p)$ and define

$$(\mathbf{B}_1|\mathbf{B}_2)=(\widetilde{\mathbf{B}}_1|\widetilde{\mathbf{B}}_2)\mathbf{R},\ (\mathbf{B}_1^*|\mathbf{B}_2^*)=(\widetilde{\mathbf{B}}_1^*|\widetilde{\mathbf{B}}_2^*)\mathbf{R}^*.$$

Observe that, the distribution of basis $(\mathbf{B}_1,\mathbf{B}_2,\mathbf{B}_3)$ is properly. We then chooses $\alpha\leftarrow\mathbb{Z}_p$, $\mathbf{w}_{1,1},\ldots,\mathbf{w}_{1,l_1},\ \mathbf{w}_{2,1},\ldots,\mathbf{w}_{2,l_2}\leftarrow\mathbb{Z}_p^{1\times(2k+1)}$, and simulate $\mathbf{Game}_{2.q}$ as follows.

When the adversary makes the key query for $\mathbf{x}=(\mathbf{x}_1,\mathbf{x}_2)$, we use $\widetilde{\mathbf{B}}_1$, $\widetilde{\mathbf{B}}_2$ to output

$$[\alpha+(x_{1,1}\cdot\mathbf{w}_{1,1}+\cdots+x_{1,l_1}\cdot\mathbf{w}_{1,l_1}+x_{2,1}\cdot\mathbf{w}_{2,1}+\cdots+x_{2,l_2}\cdot\mathbf{w}_{2,l_2})\mathbf{d}]_2,\ [\mathbf{d}]_2,$$

where $\mathbf{d}\leftarrow\mathsf{span}_c(\widetilde{\mathbf{B}}_1,\widetilde{\mathbf{B}}_2)$. Although we sample $\mathbf{d}$ using $\widetilde{\mathbf{B}}_1$, $\widetilde{\mathbf{B}}_2$, the vector is uniformly distributed over $\mathsf{span}_c(\mathbf{B}_1,\mathbf{B}_2)$ as required.

At some point, when the adversary submits the challenge $(\mathbf{y}^{(0)}=(\mathbf{y}_1^{(0)},\mathbf{y}_2),\mathbf{y}^{(1)}=(\mathbf{y}_1^{(1)},\mathbf{y}_2),m_0,m_1)$ with $m_0=m_1=m$, we outputs $y_{1,1}^{(b)}\cdot\mathbf{v}_0+y_{1,1}^{(1-b)}\cdot\mathbf{v}_1+y_{1,1}^{(b)}\cdot\mathbf{u}_1^{(3)}+\mathbf{w}_{1,1},\ \ldots,\ y_{1,l_1}^{(b)}\cdot\mathbf{v}_0+y_{1,l_1}^{(1-b)}\cdot\mathbf{v}_1+y_{1,l_1}^{(b)}\cdot\mathbf{u}_1^{(3)}+\mathbf{w}_{1,l_1},\ y_{2,1}\cdot\mathbf{u}_2+\mathbf{w}_{2,1},\ \ldots,\ y_{2,l_2}\cdot\mathbf{u}_2+\mathbf{w}_{2,l_2},\ [\alpha]_2\cdot m$, where $\mathbf{u}_1^{(3)}\leftarrow\mathsf{span}_r(\mathbf{B}_3^{*\top})$, $\mathbf{u}_2\leftarrow\mathbb{Z}_p^{1\times(2k+1)}$ and

$$\mathbf{v}_0=\mathbf{u}_1^{(1)}\leftarrow\mathsf{span}_r(\mathbf{B}_1^{*\top}),\ \mathbf{v}_1=\mathbf{u}_1^{(2)}\leftarrow\mathsf{span}_r(\mathbf{B}_2^{*\top}).$$

Observe that, we have a 2-by-$(k+1)$ matrix $\mathbf{V}$ of rank 2 such that

$$\begin{pmatrix} -\mathbf{v}_0- \\ -\mathbf{v}_1- \end{pmatrix} = \mathbf{V}(\mathbf{B}_1^*|\mathbf{B}_2^*)^\top = \mathbf{V}\mathbf{R}^{-1}(\widetilde{\mathbf{B}}_1^*|\widetilde{\mathbf{B}}_2^*)^\top.$$

Since $\mathbf{R}$ is independent of other part of simulation, $\mathbf{V}\mathbf{R}^{-1}$ are uniformly distributed over $\mathbb{Z}_p^{2\times(k+1)}$ and thus it is equivalent to sample $\mathbf{v}_0, \mathbf{v}_1 \leftarrow \mathsf{span}_r((\widetilde{\mathbf{B}}_1^*|\widetilde{\mathbf{B}}_2^*)^\top)$ when creating the challenge ciphertext. Therefore, we simulate Game_3 (with respect to $\widetilde{\mathbf{B}}_1, \widetilde{\mathbf{B}}_2, \mathbf{B}_3$) simultaneously. This proves the lemma. □

Lemma 11. Game_3 *is identical to* Game_4.

Proof. By the facts shown in Section 5.3, it is implied by the statement that, for all $\mathbf{u}_1^{(3)} \in \mathsf{span}_r(\mathbf{B}_3^{*\top})$, it holds that

$$\{y_{1,i}^{(b)} \cdot \mathbf{u}_1^{(3)} + \mathbf{w}_{1,i}^{(3)}\}_{i\in[l_1]} \equiv \{\mathbf{w}_{1,i}^{(3)}\}_{i\in[l_1]} \equiv \{y_{1,i}^{(1-b)} \cdot \mathbf{u}_1^{(3)} + \mathbf{w}_{1,i}^{(3)}\}_{i\in[l_1]},$$

when $\{\mathbf{w}_{1,i}^{(3)}\}_{i\in[l_1]} \leftarrow \mathsf{span}_r(\mathbf{B}_3^{*\top})$. Therefore, for all $\mathbf{u}_{1,0}, \mathbf{u}_{1,1} \leftarrow \mathbb{Z}_p^{1\times(2k+1)}$ it holds that

$$\begin{aligned}
&(\mathbf{w}_{1,1}\mathbf{B}_1, \ldots, \mathbf{w}_{1,l_1}\mathbf{B}_1, \mathbf{w}_{2,1}\mathbf{B}_1, \ldots, \mathbf{w}_{2,l_2}\mathbf{B}_1,\\
&y_{1,1}^{(0)} \cdot \mathbf{u}_{1,0}^{(12)} + y_{1,1}^{(1)} \cdot \mathbf{u}_{1,1}^{(12)} + y_{1,1}^{(b)} \cdot \mathbf{u}_1^{(3)} + \mathbf{w}_{1,1}, \ldots,\ y_{1,l_1}^{(0)} \cdot \mathbf{u}_{1,0}^{(12)} + y_{1,l_1}^{(1)} \cdot \mathbf{u}_{1,1}^{(12)} + y_{1,l_1}^{(b)} \cdot \mathbf{u}_1^{(3)} + \mathbf{w}_{1,l_1},\\
&y_{2,1} \cdot \mathbf{u}_2^{(1)} + y_{2,1} \cdot \mathbf{u}_2^{(2)} + y_{2,1} \cdot \mathbf{u}_2^{(3)} + \mathbf{w}_{2,1}, \ldots,\ y_{2,l_2} \cdot \mathbf{u}_2^{(1)} + y_{2,l_2} \cdot \mathbf{u}_2^{(2)} + y_{2,l_2} \cdot \mathbf{u}_2^{(3)} + \mathbf{w}_{2,l_2})\\
\equiv\ &(\mathbf{w}_{1,1}\mathbf{B}_1, \ldots, \mathbf{w}_{1,l_1}\mathbf{B}_1, \mathbf{w}_{2,1}\mathbf{B}_1, \ldots, \mathbf{w}_{2,l_2}\mathbf{B}_1,\\
&y_{1,1}^{(0)} \cdot \mathbf{u}_{1,0} + y_{1,1}^{(1)} \cdot \mathbf{u}_{1,1} + \mathbf{w}_{1,1}, \ldots,\ y_{1,l_1}^{(0)} \cdot \mathbf{u}_{1,0} + y_{1,l_1}^{(1)} \cdot \mathbf{u}_{1,1} + \mathbf{w}_{1,l_1},\\
&y_{2,1} \cdot \mathbf{u}_2^{(1)} + y_{2,1} \cdot \mathbf{u}_2^{(2)} + y_{2,1} \cdot \mathbf{u}_2^{(3)} + \mathbf{w}_{2,1}, \ldots,\ y_{2,l_2} \cdot \mathbf{u}_2^{(1)} + y_{2,l_2} \cdot \mathbf{u}_2^{(2)} + y_{2,l_2} \cdot \mathbf{u}_2^{(3)} + \mathbf{w}_{2,l_2})
\end{aligned}$$

This completes the proof. □
□

J Proof of Theorem 6

Proof (of Theorem 6). We will prove the theorem using a hybrid argument over a sequence of games. Following [12,32,10], without loss of generality, we assume that the messages m_0, m_1 submitted by the adversary at the challenge phase are equal, i.e., $m_0 = m_1 = m$, since we can reduce the case $m_0 \neq m_1$ to the case $m_0 = m_1$ by arguing that an encryption for m_0 is indistinguishable with an encryption for m_1. We also assume that the adversary makes at most q key queries.

Game_0 is the real game in which the challenge ciphertext for $\mathbf{y}^{(b)} = (\mathbf{y}_1^{(b)} = (y_{1,1}^{(b)}, \ldots, y_{1,l_1}^{(b)}), \mathbf{y}_2 = (y_{2,1}, \ldots, y_{2,l_2}))$ is of the form

$$[\mathbf{c}^\top]_1,$$

$$[\mathbf{c}^\top(y_{1,1}^{(b)} \cdot \mathbf{U}_1 + \mathbf{W}_{1,1})]_1, \ldots, [\mathbf{c}^\top(y_{1,l_1}^{(b)} \cdot \mathbf{U}_1 + \mathbf{W}_{1,l_1})]_1,$$
$$[\mathbf{c}^\top(y_{2,1} \cdot \mathbf{U}_2 + \mathbf{W}_{2,1})]_1, \ldots, [\mathbf{c}^\top(y_{2,l_2} \cdot \mathbf{U}_2 + \mathbf{W}_{2,l_2})]_1,$$
$$e([\mathbf{c}^\top]_1, [\mathbf{k}]_2) \cdot m$$

where $b \leftarrow \{0,1\}, \mathbf{c} \leftarrow \mathsf{span}_c(\mathbf{A})$.

Game$_1$ is identical to **Game**$_0$ except that we pick $\mathbf{c} \leftarrow \mathbb{Z}_p^{k+1}$ when generating the challenge ciphertext.

We prove this theorem by the following lemmas. Lemma 12 states that **Game**$_0$ and **Game**$_1$ are indistinguishable; and Lemma 13 states that the advantage of the adversary in **Game**$_1$ is negligible. Therefore, we conclude that the advantage of the adversary in **Game**$_0$ is negligible. This completes the proof.

Lemma 12. *Under the* MDDH$_k$ *assumption,* **Game**$_0$ *and* **Game**$_1$ *are computationally indistinguishable.*

Proof. Suppose there exists an algorithm $\mathcal{A}$ that distinguishes **Game**$_0$ and **Game**$_1$. Then we can build an algorithm $\mathcal{B}$ breaking MDDH$_k$ assumption with non-negligible advantage. $\mathcal{B}$ is given $[\mathbf{A}]_1$, $[\mathbf{c}]_1$ and going to tell whether $\mathbf{c} \leftarrow \mathsf{span}_c(\mathbf{A})$ or $\mathbf{c} \leftarrow \mathbb{Z}_p^{k+1}$. $\mathcal{B}$ will simulate **Game**$_0$ or **Game**$_1$ for $\mathcal{A}$. First $\mathcal{B}$ chooses $\mathbf{k} \leftarrow \mathbb{Z}_p^{k+1}$, $\mathbf{U}_1, \mathbf{U}_2, \mathbf{W}_{1,1}, \ldots, \mathbf{W}_{1,l_1}$, $\mathbf{W}_{2,1}, \ldots, \mathbf{W}_{2,l_2} \leftarrow \mathbb{Z}_p^{(k+1)\times(2k+1)}$, and $\mathbf{B}_1 \leftarrow \mathbb{Z}_p^{(2k+1)\times k}$. Then, $\mathcal{B}$ uses $[\mathbf{A}]_1$ to output $([\mathbf{A}^\top]_1, [\mathbf{A}^\top\mathbf{U}_1]_1, [\mathbf{A}^\top\mathbf{U}_2]_1, [\mathbf{A}^\top\mathbf{W}_{1,1}]_1, \ldots, [\mathbf{A}^\top\mathbf{W}_{1,l_1}]_1, [\mathbf{A}^\top\mathbf{W}_{2,1}]_1, \ldots, [\mathbf{A}^\top\mathbf{W}_{2,l_2}]_1, [\mathbf{A}^\top\mathbf{k}]_T)$.

When $\mathcal{A}$ makes the key query for $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$, $\mathcal{B}$ outputs

$$[\mathbf{k} + (x_{1,1} \cdot \mathbf{W}_{1,1} + \cdots + x_{1,l_1} \cdot \mathbf{W}_{1,l_1} + x_{2,1} \cdot \mathbf{W}_{2,1} + \cdots + x_{2,l_2} \cdot \mathbf{W}_{2,l_2})\mathbf{d}]_2,\ [\mathbf{d}]_2,$$

where $\mathbf{d} \leftarrow \mathsf{span}_c(\mathbf{B}_1)$.

At some point, when $\mathcal{A}$ submits the challenge $(\mathbf{y}^{(0)} = (\mathbf{y}_1^{(0)}, \mathbf{y}_2), \mathbf{y}^{(1)} = (\mathbf{y}_1^{(1)}, \mathbf{y}_2), m_0, m_1)$ with $m_0 = m_1 = m$, $\mathcal{B}$ outputs

$$[\mathbf{c}^\top]_1,$$
$$[\mathbf{c}^\top(y_{1,1}^{(b)} \cdot \mathbf{U}_1 + \mathbf{W}_{1,1})]_1, \ldots, [\mathbf{c}^\top(y_{1,l_1}^{(b)} \cdot \mathbf{U}_1 + \mathbf{W}_{1,l_1})]_1,$$
$$[\mathbf{c}^\top(y_{2,1} \cdot \mathbf{U}_2 + \mathbf{W}_{2,1})]_1, \ldots, [\mathbf{c}^\top(y_{2,l_2} \cdot \mathbf{U}_2 + \mathbf{W}_{2,l_2})]_1,$$
$$e([\mathbf{c}^\top]_1, [\mathbf{k}]_2) \cdot m$$

Observe that, when $\mathbf{c}$ is uniformly distributed over $\mathsf{span}_c(\mathbf{A})$, $\mathcal{B}$ has properly simulated **Game**$_0$; otherwise, when $\mathbf{c}$ is uniformly distributed over $\mathbb{Z}_p^{k+1}$, $\mathcal{B}$ has properly simulated **Game**$_1$. This proves the lemma. □

Lemma 13. *If the private-key dual IPE scheme described in Fig. 8 is adaptively secure, the advantage of the adversary in* **Game**$_1$ *is negligible.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that has non-negligible advantage in **Game**$_1$. Then we can build an algorithm $\mathcal{B}$ that makes use of $\mathcal{A}$ to break

the adaptive security of private-key dual IPE scheme described in Fig. 8 with non-negligible advantage. $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine and proceeds as follows.

First $\mathcal{B}$ chooses $(\mathbf{A}, \mathbf{c}) \leftarrow \mathbb{Z}_p^{(k+1)\times k} \times \mathbb{Z}_p^{k+1}$, $\widetilde{\mathbf{k}} \leftarrow \mathbb{Z}_p^k$, $\widetilde{\mathbf{U}}_1, \widetilde{\mathbf{U}}_2, \widetilde{\mathbf{W}}_{1,1}, \ldots, \widetilde{\mathbf{W}}_{1,l_1}$, $\widetilde{\mathbf{W}}_{2,1}, \ldots, \widetilde{\mathbf{W}}_{2,l_2} \leftarrow \mathbb{Z}_p^{k\times(2k+1)}$. Then, it sends the public key $([\mathbf{A}^\top]_1, [\widetilde{\mathbf{U}}_1]_1, [\widetilde{\mathbf{U}}_2]_1$, $[\widetilde{\mathbf{W}}_{1,1}]_1, \ldots, [\widetilde{\mathbf{W}}_{1,l_1}]_1, [\widetilde{\mathbf{W}}_{2,1}]_1, \ldots, [\widetilde{\mathbf{W}}_{2,l_2}]_1, [\widetilde{\mathbf{k}}]_T)$ to the adversary $\mathcal{A}$. Note that, the public key is simulated properly.

When $\mathcal{A}$ makes the key query for $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$, $\mathcal{B}$ forwards the query to its environment and receives (K_0, K_1). Then, $\mathcal{B}$ computes

$$\widetilde{K}_0 = [\widetilde{\mathbf{k}}]_2 \cdot ((x_{1,1} \cdot \widetilde{\mathbf{W}}_{1,1} + \cdots + x_{1,l_1} \cdot \widetilde{\mathbf{W}}_{1,l_1} + x_{2,1} \cdot \widetilde{\mathbf{W}}_{2,1} + \cdots + x_{2,l_2} \cdot \widetilde{\mathbf{W}}_{2,l_2}) \odot K_1)$$

and sends the secret key

$$(\mathbf{T} \odot \begin{pmatrix} \widetilde{K}_0 \\ K_0 \end{pmatrix},\ K_1),\ \text{where } \mathbf{T} = \begin{pmatrix} \mathbf{A}^\top \\ \mathbf{c}^\top \end{pmatrix}^{-1}$$

to the adversary $\mathcal{A}$. Notice that, with overwhelming probability, $(\mathbf{A}|\mathbf{c})$ is full-rank. If (K_0, K_1) is a private-key dual IPE secret key, secrets keys we computed is for our public-key dual IPE.

At some point, when $\mathcal{A}$ submits the challenge $(\mathbf{y}^{(0)}, \mathbf{y}^{(1)}, m_0, m_1)$ with $m_0 = m_1 = m$, $\mathcal{B}$ submits $(\mathbf{y}^{(0)}, \mathbf{y}^{(1)}, 1, 1)$ to its environment and receives $(C_{1,1}, \ldots, C_{1,l_1}$, $C_{2,1}, \ldots, C_{2,l_2}, C)$. Then, $\mathcal{B}$ sends

$$[\mathbf{c}^\top]_1,\ [C_{1,1}]_1,\ \ldots,\ [C_{1,l_1}]_1,\ [C_{2,1}]_1,\ \ldots,\ [C_{2,l_2}]_1,\ e([1]_1, C) \cdot m$$

to the adversary $\mathcal{A}$.

Finally, $\mathcal{B}$ outputs $\mathcal{A}$'s guess bit. Observe that, if $(C_{1,1}, \ldots, C_{1,l_1}, C_{2,1}, \ldots, C_{2,l_2}$, $C)$ is a private-key dual IPE ciphertext for $b = 0$, the ciphertext we created is a public-key dual IPE ciphertext for $b = 0$; this also holds for $b = 1$. This proves the lemma. □

□

K Security analysis of $\Sigma^{\mathcal{R}'}_{\mathbf{clause}}$

In this section, we show that the Sigma protocol $\Sigma^{\mathcal{R}'}_{\text{clause}}$ constructed in Fig. 10 is complete, supports knowledge soundness and is special HVZK. In fact, we can view $\Sigma^{\mathcal{R}'}_{\text{clause}}$ as a matrix version of Chaum-Pedersen protocol [8].

Completeness. Given a valid statement-witness pair $(x = (\tau_{\mathsf{p}}, c, pk, \mathsf{m}_{\mathtt{dum}}, j), w = \overline{\mathbf{s}})$ and a normally generated transcript $(\mathsf{a}, \mathsf{c}, \mathsf{z} = \mathbf{z_t})$, we have

$$\begin{aligned} \mathsf{a}'_{C_0} &= [\mathbf{z_t}^\top \mathbf{A}^\top]_1 (\mathsf{c} \odot C_0) \\ &= [(\mathbf{t} - \mathsf{c} \cdot \mathbf{s})^\top \mathbf{A}^\top]_1 (\mathsf{c} \odot C_0) \\ &= [\mathbf{t}^\top \mathbf{A}^\top]_1 ((-\mathsf{c}) \odot [\mathbf{s}^\top \mathbf{A}^\top]_1)(\mathsf{c} \odot C_0) \\ &= [\mathbf{t}^\top \mathbf{A}^\top]_1 ((-\mathsf{c}) \odot C_0)(\mathsf{c} \odot C_0) \end{aligned}$$

$$= [\mathbf{t}^\top \mathbf{A}^\top]_1$$
$$= \mathsf{a}_{C_0}$$

Similarly, we can check that $\mathsf{a}'_{C_{1,i}} = \mathsf{a}_{C_{1,i}}$ for all $i \in [l_1]$, $\mathsf{a}'_{C_{2,\iota}} = \mathsf{a}_{C_{2,\iota}}$ for all $\iota \in [l_2]$ and $\mathsf{a}'_C = \mathsf{a}_C$. Thus, $\mathcal{V}_2$ would output 1, which implies that $\Sigma'^{\mathcal{R}'}_{\text{clause}}$ is complete.

Knowledge soundness. Given two accepting transcripts $(\mathsf{a}, \mathsf{c}, \mathsf{z})$ and $(\mathsf{a}, \mathsf{c}', \mathsf{z}')$ for some statement $x = (\tau_{\mathsf{p}}, c, pk, \mathsf{m}_{\mathtt{dum}}, j)$, we can compute

$$w = (\mathsf{c}' - \mathsf{c})^{-1} \cdot (\mathbf{z} - \mathbf{z}').$$

Now we show that w is indeed a valid witness for x. Let $\mathbf{z}_{\mathbf{t}} = \mathsf{z}$ and $\mathbf{z}'_{\mathbf{t}} = \mathsf{z}'$. We can check that

$$\begin{aligned}
[w^\top \mathbf{A}^\top]_1 &= (\mathsf{c}' - \mathsf{c})^{-1} \odot [(\mathbf{z}_{\mathbf{t}} - \mathbf{z}'_{\mathbf{t}})^\top \mathbf{A}^\top]_1 \\
&= (\mathsf{c}' - \mathsf{c})^{-1} \odot [(\mathbf{z}_{\mathbf{t}}^\top \mathbf{A}^\top - (\mathbf{z}'_{\mathbf{t}})^\top \mathbf{A}^\top]_1 \\
&= (\mathsf{c}' - \mathsf{c})^{-1} \odot ([(\mathbf{z}_{\mathbf{t}}^\top \mathbf{A}^\top]_1 / [(\mathbf{z}'_{\mathbf{t}})^\top \mathbf{A}^\top]_1) \\
&= (\mathsf{c}' - \mathsf{c})^{-1} \odot ((\mathsf{a}_{C_0} / (\mathsf{c} \odot C_0)) / (\mathsf{a}_{C_0} / (\mathsf{c}' \odot C_0))) \quad /\!/ \text{ correctness} \\
&= (\mathsf{c}' - \mathsf{c})^{-1} \odot ((\mathsf{c}' - \mathsf{c}) \odot C_0)) \\
&= C_0
\end{aligned}$$

Thus, w is the witness for C_0. Similarly, we can check that w is the witness for other components in ciphertext c. Thus, w is a valid witness for $x = (\tau_{\mathsf{p}}, c, pk, \mathsf{m}_{\mathtt{dum}}, j)$. It implies that $\Sigma'^{\mathcal{R}'}_{\text{clause}}$ supports knowledge soundness.

Special HVZK. We construct a simulator Sim as shown in Fig. 19.

$\underline{\mathsf{Sim}(x, \mathsf{c})}$:
$(\tau_{\mathsf{p}}, c, pk, \mathsf{m}_{\mathtt{dum}}, j) \leftarrow x$, $\mathbf{z}_{\mathbf{t}} \leftarrow \mathbb{Z}_p$, $\mathsf{z} \leftarrow \mathbf{z}_{\mathbf{t}}$
$\mathsf{a}_{C_0} \leftarrow [\mathbf{z}^\top \mathbf{A}^\top]_1 (\mathsf{c} \odot C_0)$, $\mathsf{a}_{C_{1,j}} \leftarrow [\mathbf{z}_{\mathbf{t}}^\top \mathbf{A}^\top (\mathbf{U}_1 + \mathbf{W}_{1,j})]_1 (\mathsf{c} \odot C_{1,j})$
For each $i \in [l_1] \backslash \{j\}$: $\mathsf{a}_{C_{1,i}} \leftarrow [\mathbf{z}_{\mathbf{t}}^\top \mathbf{A}^\top \mathbf{W}_{1,i}]_1 (\mathsf{c} \odot C_{1,i})$
For each $\iota \in [l_2]$: $\mathsf{a}_{C_{2,\iota}} \leftarrow [\mathbf{z}_{\mathbf{t}}^\top \mathbf{A}^\top ((\tau_{\mathsf{p}})_\iota \cdot \mathbf{U}_2 + \mathbf{W}_{2,\iota})]_1 (\mathsf{c} \odot C_{2,\iota})$
$\mathsf{a}_C \leftarrow [\mathbf{z}_{\mathbf{t}}^\top \mathbf{A}^\top \mathbf{k}]_T \cdot (\mathsf{c} \odot (C / \mathsf{m}_{\mathtt{dum}}))$
Return $(\mathsf{a} \leftarrow (\mathsf{a}_{C_0}, (\mathsf{a}_{C_{1,i}})_{i \in [l_1]}, (\mathsf{a}_{C_{2,\iota}})_{\iota \in [l_2]}, \mathsf{a}_C), \mathsf{z})$

Fig. 19: Algorithms of simulator Sim for $\Sigma'^{\mathcal{R}'}_{\text{clause}}$

Consider the transcript $(\mathsf{a}, \mathsf{c}, \mathsf{z})$, where the commitment and the response (a, z) are output by the simulator. It is clear that $(\mathsf{a}, \mathsf{c}, \mathsf{z})$ is an accepting transcript for x. Now, we show that it has the same distribution as that of a transcript bewteen $\mathcal{P}(x, w)$ and $\mathcal{V}(x)$.

1. First of all, $\mathsf{z} = \mathbf{z_t}$ output by the simulator is uniformly distributed over $\mathbb{Z}_p$. Note that in $\mathcal{P}_2$, $\mathbf{z_t} = \mathbf{t} - \mathsf{c} \cdot \mathbf{s}$. Since $\mathbf{t}$ is sampled randomly over $\mathbb{Z}_p$, we can know that $\mathsf{z} = \mathbf{z_t}$ output by $\mathcal{P}_2$ is also uniformly distributed over $\mathbb{Z}_p$.
2. The challenges c in both cases are randomly over $\mathbb{Z}_p$. Note that c and z are uniformly and independently distributed over $\mathbb{Z}_p$.
3. Then, we consider a_{C_0} output by the simulator and that output by $\mathcal{P}_1$.
   - With respect to the a_{C_0} output by the algorithm of Sim, it holds that $\mathsf{a}_{C_0} = [\mathbf{z_t}^\top \mathbf{A}^\top]_1(\mathsf{c} \odot C_0)$.
   - With respect to the a_{C_0} output by the algorithm of $\mathcal{P}_1$, the correctness guarantees that $\mathsf{a}_{C_0} = \mathsf{a}'_{C_0} = [\mathbf{z_t}^\top \mathbf{A}^\top]_1(\mathsf{c} \odot C_0)$.

   Therefore, in both algorithms, $\mathsf{a}_{C_0} = [\mathbf{z_t}^\top \mathbf{A}^\top]_1(\mathsf{c} \odot C_0)$, determined by the responses $\mathbf{z_t}$ and the challenge c. Since $\mathbf{z_t}$ and c in both cases are uniformly and independently distributed over $\mathbb{Z}_p$, we can conclude that a_{C_0} output by the simulator has the same distribution as that output by the prover $\mathcal{P}$.
4. Similarly, we can analyze that $((\mathsf{a}_{C_{1,i}})_{i\in[l_1]}, (\mathsf{a}_{C_{2,\iota}})_{\iota\in[l_2]}, \mathsf{a}_C)$ output by the simulator has the same distribution as those output by the prover $\mathcal{P}$.

So $(\mathsf{a}, \mathsf{c}, \mathsf{z})$ output by the simulator has the same distribution as that of a transcript between $\mathcal{P}$ and $\mathcal{V}$, which implies that the protocol $\Sigma^{\mathcal{R}'}_{\text{clause}}$ is special HVZK.

L Proof of Theorem 7

Proof (of Theorem 7). We first prove that $\Sigma^{\mathcal{R}''}_{\text{clause}}$ is a Sigma protocol for ψ-preimages [14]. Then, according to Theorem 9 (in Appendix A.3), $\Sigma^{\mathcal{R}''}_{\text{clause}}$ is stackable.

For $\Sigma^{\mathcal{R}''}_{\text{clause}}$, we set $\mathfrak{G}_1 = \mathbb{Z}_p^2$, $\mathfrak{G}_2 = \mathbb{G}_{\mathsf{com}} \times \mathbb{G}_1^{3(l_1+l_2)+2} \times \mathbb{G}_T$ and $\mathcal{CL} = [0, p)$, where $\mathbb{G}_{\mathsf{com}}$, $\mathbb{G}_1$ and $\mathbb{G}_T$ are cyclic groups of prime order p. We construct a function $\psi : \mathfrak{G}_1 \to \mathfrak{G}_2$ as follows:

$$\begin{aligned}\psi(w = (r_{\text{com}}, \mathbf{s})) := (\ & h^{r_{\text{com}}}, [\mathbf{s}^\top \mathbf{A}^\top]_1, [\mathbf{s}^\top \mathbf{A}^\top \mathbf{W}_{1,1}]_1, \ldots, [\mathbf{s}^\top \mathbf{A}^\top \mathbf{W}_{1,j-1}]_1), \\ & [\mathbf{s}^\top \mathbf{A}^\top (\mathbf{U}_1 + \mathbf{W}_{1,j})]_1, [\mathbf{s}^\top \mathbf{A}^\top \mathbf{W}_{1,j+1}]_1, \ldots, [\mathbf{s}^\top \mathbf{A}^\top \mathbf{W}_{1,l_1}]_1, \\ & [\mathbf{s}^\top \mathbf{A}^\top ((\tau_{\mathsf{p}})_1 \cdot \mathbf{U}_2 + \mathbf{W}_{2,1})]_1, \ldots, [\mathbf{s}^\top \mathbf{A}^\top ((\tau_{\mathsf{p}})_{l_2} \cdot \mathbf{U}_2 + \mathbf{W}_{2,l_2})], \\ & [\mathbf{s}^\top \mathbf{A}^\top \mathbf{k}]_T \cdot \mathsf{m}_{\mathtt{dum}}\ ).\end{aligned}$$

Therefore, the $\Sigma^{\mathcal{R}''}_{\text{clause}}$ is a Sigma protocol for ψ-preimages. According to Theorem 9, $\Sigma^{\mathcal{R}''}_{\text{clause}}$ is stackable. □