Chosen-Ciphertext Security for Inner Product FE: Multi-Client and Multi-Input, Generically

Ky Nguyen

DIENS, École normale supérieure, CNRS, Inria, PSL University, Paris, France

Abstract. Functional Encryption is a powerful cryptographic primitive that allows for fine-grained access control over encrypted data. In the multi-user setting, especially Multi-Client and Multi-Input, a plethora of works have been proposed to study on concrete function classes, improving security, and more. However, the CCA-security for such schemes is still an open problem, where the only known works are on Public-Key Single-Client FE (*e.g.* Benhamouda, Bourse, and Lipmaa, PKC'17). This work provides the first generic construction of CCA-secure Multi-Client FE for Inner Products, and Multi-Input FE for Inner Products, with instantiations from SXDH and DLIN for the former, and DDH or DCR for the latter. Surprisingly, in the case of secret key MIFE we attain the same efficiency as in the public key single-client setting. In the MCFE setting, a toolkit of CCA-bootstrapping techniques is developed to achieve CCA-security in its *secret key* setting.

1 Introduction and Motivation

Functional Encryption (FE). Secure communication fundamentally uses encryption schemes as its building block: only with a decryption key can a recipient decrypt a ciphertext to obtain the underlying message, otherwise it is guaranteed by the scheme's semantic security that nothing is leaked about the plaintext data. This *all-or-nothing* nature remains the standard for a long time. However, the fact that more complex and hierarchical comunication systems appear means more need of a fine-grained control over the leakage from ciphertexts. Thus a great deal of motivation into advanced notions of encryption is generated. The progress culminates in Functional Encryption (FE) [76, 23] that is introduced by Boneh, Sahai and Waters. FE allows a finer treatment of what a recipient can obtain: each decryption key is associated with a function and the decryption result *as per* this key is guaranteed to reveal no more than the foregoing function evaluation on the underlying plaintext. In principle, these *functional decryption keys* permit controlling the amount of information to be at most the functional analysis over the plaintext, and not more.

FE received large interest from the cryptographic community, first as a generalization of Identity-Based Encryption (IBE) [77, 33, 21, 22] and Attribute-Based Encryption (ABE) [76, 47, 73, 16, 72], where the latter two unfortunately provide only access control over decryption keys while retaining an all-or-nothing result. Realizing FE for concrete function classes is first done by Abdalla *et al.* [5], where a functional decryption key allows decrypting to an inner product between some function vector and the plaintext vector. A such FE scheme is coined *inner product* FE (IPFE). A long line of works on IPFE spans over almost a decade, with numerous interesting results, to improve existing constructions for inner products [11, 18, 27] to move to quadratic functions [17, 42, 15, 61], or to concoct new advanced notions [45] as well as relate to other notions in cryptography [14, 19]. In particular, great enthusiast is manifested in the *multi-user* setting [31, 3, 2, 59, 32, 9, 78, 66, 69]. This setting allows multiple users to take part in an FE system and contribute either to some joint ciphertext or joint functional key. Aggregation of contributed keys will be able to decrypt aggregation of contributed ciphertexts upon certain conditions, *e.g.* those partial keys share identical function tags and those partial ciphertexts share identical timestamps. A number of different FE notions can stem from different flavors of security, with or without function privacy, regarding corruption of users' keys, as well as flexibility in the number of users. At first sights this setting seems similar to multi-party computation (MPC), where several players provide their inputs for a jointly function evaluation while maintaining their input privacy. However, the principal difference is found in the fact that FE is expected to be non-interactive in both encryption and decryption, and is thus more preferable in decentralized or dynamic environments. While FE with a single encryptor might be of theoretical interest, in real-life, the number of really useful functions may be limited. When this number of functions is small, any PKE can be converted into FE by additionally encrypting the evaluations by the various functions under specific keys. This approach is impossible for multiple users, even when a unique fixed function is considered.

The center of attention of this paper is FE schemes in the multi-user setting for concrete function classes such as inner products.

More on Multi-User Settings. Taking a closer look at how to meaningfully define FE in the multi-user setting, it appears that inherent definitional changes are necesary. Naively, as mentioned earlier, when the number of useful functions is polynomially large, Public Key Encryption (PKE) can be used to encrypt each function evaluation using a different public key. Unfortunately in the multi-user setting where evaluating a function requires different inputs from different users, this PKE-based approach is impossible as the result is unknown at encryption time, even when there is only one function. A systematic study into Multi-Input Functional Encryption (MIFE) and Multi-Client Functional Encryption (MCFE) is then conducted in [44, 46]. Both MIFE and MCFE permit decrypting to a function evaluation over a list of inputs. In MIFE, a *single encryptor* can encrypt different inputs in the list at different time, whereas in MCFE there are multiple *clients* who independently encrypt their respective input. A trusted authority is required to issue functional keys to decrypt jointly the ciphertexts. Last but not least, due to possible combination of ciphertexts for decryption, both MIFE and MCFE are defined as secret-key primitives so as to have non-trivial security guarantees.

Technically, in the case of MCFE, an index i for each client and a (typically time-based) tag tag are used for every encryption: $(c_1 = \mathsf{Enc}(1, x_1, \mathsf{tag}), \ldots, c_n = \mathsf{Enc}(n, x_n, \mathsf{tag}))$. Anyone owning a functional decryption key dk_f, for an n-ary function f and multiple ciphertexts (for the same tag tag, in the case of MCFE) can compute $f(x_1, \ldots, x_n)$ but nothing else about the individual x_i's. Implicitly, clients have to be able to coordinate together on the tags, and different usability in practice. In particular, in MCFE, the combination of ciphertexts generated for different tags does not give a valid global ciphertext and the adversary learns nothing from it. This leads to more versatility since encrypting x_i under tag has a different meaning from encrypting x_i under $\mathsf{tag}' \neq \mathsf{tag}$. On the other hand, MIFE does not use tags and once a ciphertext of x_i is computed, it can be reused for different combinations. However, in both situations of $\mathsf{MIFE}/\mathsf{MCFE}$, we recall that encryption must require a private key, otherwise anybody could complete the vector initiated by a user in many ways, and then obtain many various evaluations from a unique functional decryption key. But then, since encryption needs a private key per client in MCFE, for each ciphertext component c_i, some of these keys might get corrupted. The focal point of this paper is narrowed down to MCFE and its brother notion MIFE.

Repetitions under One Message Tag. It should be clear from our discussion so far that the functionality of MCFE requires ciphertexts to share some identical tag so that they can be combiningly decrypted. From a security standpoint, an almost immediate question is what kind of properties we need regarding these tags. Initial works and follow-ups on MCFE [30, 59] ignore repeated adversarial queries to some slot i under the same tag tag. It is argued in [30] that it is up to the user's responsibility not to use the same mesage and key tag twice, neither for encryption nor for key-generation, respectively. Nonetheless, we believe that proving security under a repeated usage of tags is still important. First of all, mistakenly or maliciously re-using of tags can happen in practice. Furthermore, given a MCFE that is provably secure even when an adversary can obtain different ciphertexts on the same (i, tag), one can get a MIFE [44, 46] for the same function class whose semantic security is preserved from the MCFE. In short, the MIFE fixes a public tag and runs the algorithms of the MCFE using that public tag for all encryption/decryption. Any MIFE adversary that tries to combine MIFE ciphertexts are essentially some MCFE adversary who tries to break the semantic security using repeated ciphertexts under the fixed public tag. This connection is studied in preceding works [3, 2] and recently confirmed in [12, 67]. Our final aim for the security of MCFE in this work will allow repetitions on message tags.

Chosen-Ciphertext Security for Functional Encryption. As with the case of PKE or IBE/ABE, the standard IND-CPA security model for MCFE protects first and foremost against eavesdropping attacks. However, against more powerful active adversaries, the required security guarantee is a security against chosen-ciphertext attacks (IND-CCA). In short, what we expect from an IND-CCA security for FE and its multi-user variants is that a decryption oracle is not helpful to the adversary in breaking the semantic security. Defining and achieving IND-CCA security has witnessed a long line of works in PKE starting with [63, 34] and more, or in IBE/ABE [21, 56, 82].

To the best of our knowledge, the ony existing works on IND-CCA for FE are [62, 18, 28]. The work of [62] constructs IND-CCA secure public-key single client FE from IND-CPA secure FE while basing on various properties that are met by many FE schemes: key-policy or ciphertext-policy ABE, or FE for regular languages. However, their techniques do not seem to apply to the case of inner products. On the other hand, the work of [18] constructs IND-CCA secure public-key single client FE for inner products, where a complete n-long encryption contains ciphertext components for scalars $(x_i)_{i\in[n]}$ that can be decrypted by a functional key associated with a vector $\mathbf{y} \overset{def}{=} (y_i)_{i\in[n]}$. The technique of [18] circumvents the usage of *non-interactive zero-knowledge proofs* (NIZK) and also avoids the reliance on bilinear groups or the *random oracle model* (ROM), all of which are required by more general CPA-to-CCA approach such at the Naor-Yung paradigm [63]. A follow-up work [28] improves the construction of [18] by providing much more efficient schemes under the same blueprint, achieving adaptive IND-CCA security while staying public key single client FE.

Why Examining IND-CCA in Multi-User FE? Not long after the seminal works that introduce FE [76, 23], extensions of FE into the multi-user setting are initiated by Goldwasser *et al.* [44, 46] with *Multi-Client Functional Encryption* (MCFE) and *Multi-Input Functional Encryption* (MIFE). Speaking of the concrete class to compute inner products, efforts are made over the years for more refinements in definitional frameworks, efficiency, and more [37, 30, 31, 6, 3, 2, 59, 32, 7, 65]. Going beyond inner products, a rising level of interest is directed towards quadratic functions [8, 74, 10] or attribute-weighted sums [12, 68, 64].

This plethora of works on FE in the multi-user setting is a clear indication of the importance of this notion. And even until recently, new results still appear to refine the security model of MCFE/MIFE. For instance, the work of [66] goes all over again the *admissible condition* from [30] and follow-up works, that is, an adversary can only query challenges $x_i^{(0)} = x_i^{(1)}$ in case of a corrupted i. It is worth noting that under this condition, *deterministic* encryption is potentially secure. It is then showed in [66] that a weaker condition is possible, *i.e.* more attacks can be admissible. More specifically, following [66], if each client in MCFE encrypts a vector $\mathbf{x}_i$ and a function F having parameters $(\mathbf{y}_i)_i$ computes $\sum_i \langle \mathbf{x}_i, \mathbf{y}_i \rangle$, the less restrictive admissibility condition states:

for any corrupted i, the adversary can ask functions $(\mathbf{y}_i)_i$ and challenges $(\mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)})$ as long as $\langle \mathbf{x}_i^{(0)} - \mathbf{x}_i^{(1)}, \mathbf{y}_i \rangle = 0$.

One important of a such strengthening in the security model is that: under the stronger model of [66], the *deterministic* encryption is no longer secure and *probabilistic* encryption is necessary. Interestingly, a follow-up [67] of [66] shows that an IND-CPA secure MCFE under admissble condition from [66] implies IND-CPA secure public-key FE (hence includes clasisical public key encryption notions such as PKE, IBE, ABE). Looking back at the progress on IND-CCA for PKE, [79] argues that deterministic encryption is not secure against chosen-ciphertext attacks, because they are not even IND-CPA secure, and around that point of time the launch of studies on IND-CCA for PKE begins. Recent advancements [67, 66] on the security model of MCFE/MIFE are then a clear suggestion that a step further to refine the security model of MCFE/MIFE is to examine the IND-CCA security, given now that the admissible condition is more relaxed with clear connections to the IND-CPA security of FE/ABE/IBE/PKE. Leaving IND-CCA unresolved for MCFE/MIFE leaves some scenario where questions on malleability of ciphertexts are not answered, *e.g.* what if an adversary make use of a corrupted ek_i in an MCFE so as to malleate a ciphertext and learn about its contents?

1.1 Our Contributions

From the above introductory discussion, a natural research question arises:

Regarding useful and non-trivial function class such as sum of inner products, how can one define and construct IND-CCA secure MCFE/MIFE schemes?

This work is the first to embarks on this question on both the definitional aspects and the construction aspects:

1. We provide a formal definition of IND-CCA security for MCFE/MIFE scheme, with respect to general function classes, such as function classes with public inputs [67] that captures fine-grained access control over FE keys, and also the function class that computes the sum of inner products[1]. The definitions (see Definition 29, Definition 30) are based on the strong admissible condition from [66], with conceptual sanity check to ensure that our IND-CCA security definition is meaningfully stronger than IND-CPA security for MCFE/MIFE schemes (see Lemma 32).
2. Regarding MCFE, we provide a generic bootstrapping from IND-CPA secure MCFE scheme to IND-CCA secure MCFE scheme for the function class that computes functions with public inputs (see Section 4.1). Along the way, we make use of simulation sound NIZK and commitments that satisfy equivocable-extractable property. Our generic transformation also bears a technique to deal with the simulaion of the decryption oracle in the IND-CCA security game, as well as how to ensure coherence uses of *secret* encryption keys in MCFE/MIFE by commitments that are compatible with NIZK proofs. The latter is new key challenge, in comparison to all previous works that focus only on the *public-key* IND-CCA secure FE. We instantiate our generic construction for the function class for sums of inner products, under SXDH and DLIN. Theses results can be found in Section 4.
3. Regarding MIFE, we provide a generic construction of IND-CCA secure MIFE scheme for sums of inner products, based on *smooth projective hash functions* (SPHF). Our transformation differs from existing approach from public key FE for inner products [18, 28]. Firstly, we identify the need of a stronger smoothness property for SPHF. Secondly, we develop a new technique of using SPHF with key-homomorphism property (that is met by our building blocks) to achieve adaptive security for MIFE. We instantiate our generic construction for the function class for sums of inner products, under DDH or DCR, and interestingly these adaptively secure instantiations are as efficient as the selectively secure IND-CCA public key FE scheme from [18, 28], in terms of asymptotic total communication over all slots (see Remark 40). These results can be found in Section 5.

All of the above results are extensively discussed at conceptual level in Section 1.2.

[1] We can of course generalize the definitions to general function classes, but in this work we focus on efficient and concrete constructions later on.

1.2 Technical Overview

We present an oveview of our technical contents. We start by the defnitional choices that are made in our Definition 29, Definition 30 for IND-CCA security of MCFE. Then, we highlight the generic constructions for MCFE in Section 4.1 and the instantiations for inner products[2] in Section 4.2. In the third part of this overview, we discuss the generic construction for inner product MIFE in Section 5.1 and the instantiations for MIFE in Section 5.2.

Definitions and Security Notions. We start by defining the chosen-ciphertxt security model for FE in the multi-user setting, notably in the setting of MCFE and MIFE. As a reminder, the standard IND-CPA security model for MCFE/MIFE [44, 46, 30] consists of a security experiment between a challenger and an adversary, where the adversary is given access to a number of oracles to which they can make queries. The **Initialise** can be asked on $(1^\lambda, 1^n)$ to generate $(\mathsf{pp}, \mathsf{msk}, (\mathsf{ek}_i)_{i\in[n]})$, a challenge bit $b \overset{\$}{\leftarrow} \{0,1\}$ is also sampled by the challenger, with respect to the security parameter λ and the number of users n, the public parameters are then given to the adversary. As MCFE/MIFE are secret-key primitives, there is an **Enc** oracle to which any query of form (i, x_i), with tag if in case of MCFE, can be submitted for the adversary to obtain the ciphertext $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, x_i)$ (or $\mathsf{Enc}(\mathsf{ek}_i, x_i, \mathsf{tag})$ if MCFE). The challenge oracle **LoR** receives a query of form $(i, x_i^{(0)}, x_i^{(1)})$, with tag if in case of MCFE, then returns the challenge ciphertext $\mathsf{ct}_i^{(b)}$ corresponding to the challenge bit b. In order to generate functional keys of their choices, the adversary can submit F to an **Extract** oracle, which uses the description of the function F to generate the functional key dk_F and returns dk_F to the adversary. Last but not least, a stronger security notion, of MCFE in particular, allows *corruption*, *i.e.* the adversary can submit i to a **Corrupt** oracle to obtain the secret encryption key ek_i. This ability to corrupt makes the adversary stronger and thus leads to a stronger notion of security, as highlighted in the seminal work on MCFE [30] and follow-up works to enhance MCFE/MIFE with a more fine-grained corruption model [9, 66]. Finally, the adversary outputs a guess bit b', and the experiment is considered successful if $b' = b$. In the following we will consider the presence of **Corrupt** oracle in the security model for MCFE. We will say explicitly when moving to MIFE and do not include **Corrupt**.

A Need for Strong Admissibility in MCFE for IND-CCA Security. As a generalization of IBE and ABE, the functional nature of FE, *i.e.* allowing evaluations on plaintexts by decrypting ciphertexts, begs a care in the security model. The best one can hope for is a notion that protects against adversaries who exploit only the functionalities of the function class, namely exclude only *trivial* attacks, nothing more. In other words, the less strict constraints we put on the adversary, the stronger the security notion is. In the case of MCFE with corruption, from the seminal paper [30], to the best of our knowledge, to other follow-up studies on MCFE (or its *decentralized* version), *e.g.* [31, 3, 2, 59, 32, 9], an *admissibility condition* is administered in order to prohibit trivial attacks, and restricted particularly adversaries to asking the challenge components $x_i^{(0)} = x_i^{(1)}$ in case of a corrupted i. This admissibility condition, however, allows *deterministic* encryption for MCFE and to some extents, this does not suffice for IND-CCA security. A classical argument that deterministic encryption is not secure against chosen-ciphertext attacks as they are not even IND-CPA secure, as argued in [79] in the case of PKE. In the context of MCFE, even though being secret-key, the same argument applies: given a deterministic encryption scheme, under some form of homomorphic property over the ciphertexts (which is satisfied by most ElGamal-based MCFE such as [30] and many more), an adversary can malleate the ciphertexts to obtain a valid decryption under a function of their choice.

A recent work [66] revisits this so far standard admissibility condition in MCFE, proposes a *less restrictive* admissibility condition, and up to certain natural function classes, the authors

[2] Throughout this paper, by "inner product" computation we mean the function class $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$ in Definition 27.

show that one cannot go weaker than their condition. With respect the function class to compute inner products of vectors, *i.e.* each client in MCFE encrypts a vector $\mathbf{x}_i$ and a function F having parameters $(\mathbf{y}_i)_i$ computes $\sum_i \langle \mathbf{x}_i, \mathbf{y}_i \rangle$, the less restrictive admissibility condition is that for any corrupted i, the adversary can ask functions $(\mathbf{y}_i)_i$ and challenges $(\mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)})$ as long as $\langle \mathbf{x}_i^{(0)} - \mathbf{x}_i^{(1)}, \mathbf{y}_i \rangle = 0$. A very recent work [67] extends the inner product schemes from [66] to integrate access control over functional keys[3], and more importantly, the authors of [67] show that the improved security model in [66] necessitates *probabilistic* encryption, and more:

> *(From [67]) an IND-CPA secure MCFE under admissble condition from [66] implies IND-CPA secure public-key FE.*

This is a strong indication that, in order to necessarily protect against chosen-ciphertext attacks in MCFE, at least the admissibility condition form [66] should be used. In our formal Definition 30 of IND-CCA and Definition 29 of admissible adversaries, we employ the admissibility condition from [66]. As a sanity check for our model, in Lemma 32, we show that with respect to our Definition 30, under corruption and strong admissiblity from [66], taking into account the IND-CCA notion for public-key FE from [18],

> *an IND-CCA secure secret-key MCFE scheme,* as per *Definition 30, gives an IND-CCA secure public-key FE (*as per *[18]).*

Combining with the definitional observation of [67] regarding IND-CPA, together with the fact that IND-CCA security is strictly stronger than IND-CPA security in PKE (as a special case of FE), our IND-CCA security model for MCFE is necessarily as least as strong as the latest IND-CPA security of MCFE so far from [66, 67].

Towards Modeling Chosen-Ciphertext Attacks: The Decryption Oracle. In the regime of FE that generalizes PKE and IBE/ABE, the work [18] initiates the IND-CCA security model for *public-key* single-client FE, then studies concrete construction to compute $\sum_{i=1}^{n} x_i y_i$ given $(\mathsf{ct}_i)_i$ of $(x_i)_i \in \mathbb{Z}_q^n$ encrypted under a public key and given $\mathsf{dk}_{(y_i)_i}$ as the functional key. Without going into the details of the IND-CCA model from [18], we observe a principle that: in order to capture chosen-ciphertext attacks in FE, necessarily the adversary should be able to decrypt the ciphertexts *with respect to functions* of their choices.

At first glance, a natural question is raised: why does a such **Dec** oracle help enhancing the security model, when the adversary is already having **Extract** to generate decryption functional keys? Coming back to practical scenario where chosen-ciphertext attacks are relevant, one might think of protecting encrypted data from being malleated *and/or* from being decrypted under functions that are not available as functional keys:

- The fact that an adversary can specify the ciphertext to query for decryption, under the constraint that they *cannot* ask on challenge ciphertexts components $\mathsf{ct}_i^{(b)}$, will capture the non-malleability.
- In order to cover the attacks that use decryption under functions that are not available as functional keys, the fact that the function can be *not* given in functional keys hints that it is not necessary to have constraints over functions in the **Dec** oracle.

Alternatively, allowing challenge ciphertext to **Dec** restrains $\hat{F}$ that the adversary can query to **Extract**:

- Initially, for *each* i,

$$\hat{F}(x_i^{(0)} \text{ combined with others by adversary}) = \hat{F}(x_i^{(1)} \text{ combined with others by adversary})$$

[3] The work [67] modelizes this access control via function classes that allow public inputs from clients at the time of encryption. These public inputs are not encrypted and are intended as the only information that is leaked, and not more. We give syntax and constructions for MCFE with respect to this function class, so as to possibly cover access control, and go beyond inner products.

- Generalized constraints when combining *multiple* challenge components decrease significantly the number of allowed functions (considering inner products' linearity).

This **Dec**-constrained model allows fewer functions, but some more queries. It might be incomparable to our current **Dec**-free model. Therefore, we only add one more admissible condition in view of the new **Dec** oracle in our Definition 30 for CCA security of MCFE: *(i)* there is no decryption query that contains some challenge ciphertext components $\mathsf{ct}_i^{(b)}$, in addition to *(ii)* the already IND-CPA improved admissibility condition from [66].

Final Thoughts in our IND-CCA Security and Towards Concrete Constructions. One quick sanity check for our model to make sure it is meaningfully stronger than the IND-CPA security model for MCFE from [66] is the following attack:

- An adversary asks for challenge ciphertexts on $(i, x_i^{(0)}, x_i^{(1)}, \mathsf{tag})$ to obtain $\mathsf{ct}_i^{(b)}$.
- The adversary malleates the challenge ciphertexts to obtain $\mathsf{ct}_i^{(\mathrm{mal},b)} \neq \mathsf{ct}_i^{(b)}$, whose underlying plaintext are *not* necessarily known to the adversary (since b stays unknown as the challenge bit).
- They then ask for a functional key on a function F such that

  $$F(x_1^{(0)}, \ldots, x_{i-1}^{(0)}, \underbrace{x_i^{(\mathrm{mal},0)}}_{\text{malleated}}, x_{i+1}^{(0)}, \ldots, x_n^{(0)}) \neq F(x_1^{(1)}, \ldots, x_{i-1}^{(1)}, \underbrace{x_i^{(\mathrm{mal},1)}}_{\text{malleated}}, x_{i+1}^{(1)}, \ldots, x_n^{(1)}) \enspace ,$$

  where $x_i^{(\mathrm{mal},b)}$ is the plaintext of $\mathsf{ct}_i^{(\mathrm{mal},b)}$, while

  $$F(x_1^{(0)}, \ldots, x_{i-1}^{(0)}, x_i^{(0)}, x_{i+1}^{(0)}, \ldots, x_n^{(0)}) = F(x_1^{(1)}, \ldots, x_{i-1}^{(1)}, x_i^{(1)}, x_{i+1}^{(1)}, \ldots, x_n^{(1)}) \enspace . \tag{1}$$

- The adversary decrypts using dk_F on the ensemble of challenge ciphertexts that contains $\mathsf{ct}_i^{(\mathrm{mal},b)}$, then decides.

There exists no IND-CPA secure MCFE scheme that can resist this attack, even in the presence of the admissibility condition from [66] (we do not have to corrupt any i), due to the choice of F satisfying Equation (1) that makes dk_F conform to the admissibility. Meanwhile, it is without difficulty to see that security in our IND-CCA model will prevent this attack: we can translate the attack into an IND-CCA adversary that queries the **Dec** oracle on $\mathsf{ct}_i^{(\mathrm{mal},b)}$, together with the function F. We emphasize that the attack is admissible in both the IND-CPA model from [66] and our IND-CCA model.

Finally, translating the foregoing IND-CCA admissible condition for the function class $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$ to compute sum of inner products of vectors, in the setting of MCFE, this gives concrete conditions below:

1. (from our Definition 29) for any challenge $(i, \mathbf{x}_i^{(0,j)}, \mathbf{x}_i^{(1,j)}, \mathsf{tag})$ to **LoR** whose response is $\mathsf{ct}_i^{(b)}$, there is no decryption query that contains $\mathsf{ct}_i^{(b)}$.
2. (from [66]) For any corrupted i, for any function $(\mathbf{y}_i)_i$ to **Extract**, for any challenges $(i, \mathbf{x}_i^{(0,j)}, \mathbf{x}_i^{(1,j)}, \mathsf{tag})$ to **LoR** up to their j-th repetition[4], it holds that $\langle \mathbf{x}_i^{(0,j)} - \mathbf{x}_i^{(1,j)}, \mathbf{y}_i \rangle = 0$.
3. (from [66]) for any function $(\mathbf{y}_i)_i$ to **Extract**, for any challenges $(\mathbf{x}_i^{(0,j)}, \mathbf{x}_i^{(1,j)})$ to **Enc** up to their j-th repetition (and for any challenge tag if in case of MCFE), it holds that $\sum_{\text{honest } i} \langle \mathbf{x}_i^{(0,j)} - \mathbf{x}_i^{(1,j)}, \mathbf{y}_i \rangle = 0$.

The above conditions will be used ubiquitously in our constructions and proofs for MCFE[5] for the inner product function class $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$.

[4] Allowing repetitions of challenge queries for fixed (i, tag) in MCFE security model is important to capture MIFE as a particular case. This is confirmed in recent works, both for standard IND-CPA and extended IND-CPA with function-hiding [31, 67]

[5] When moving to MIFE, there is no corruption and we remove the constraint 2.

MCFE Generic Constructions: Bootstrapping CPA to CCA. We now move on to our generic constructions for MCFE that attains IND-CCA security, for the general function class that allows public inputs, following Definition 26. Specifically, each client can use their secret key ek_i to encrypt (x_i, z_i) where x_i is their secret inputs and z_i is their public inputs. In MCFE, the intervention of tags at the time of encryption can be encompassed by $z_i \in \mathsf{Tag} \times \widetilde{Z}_i$, over some tag space Tag and some public input space $\widetilde{Z}_i$.

Main Ingredients for IND-CCA Security. Let us summarize the main ingredients for our generic constructions. Our starting point is $\mathsf{MCFE}^{\mathsf{cpa}}[\mathcal{F}, (\mathcal{Z}_{\lambda,i})_{i\in[n]}]$ that is IND-CPA secure for the function class $\mathcal{F}$ and the public input spaces $(\mathcal{Z}_{\lambda,i})_{i\in[n]}$. Our goal is to upgrade the IND-CPA security to IND-CCA security. We follow the natural idea to use *Non-Interactive Zero-Knowledge* (NIZK) proofs following thhe celebrated approach of Naor-Yung/Sahai [63, 75]. That is, given a $\mathsf{MCFE}^{\mathsf{cpa}}$ encryption

$$\mathsf{ct}^{\mathsf{cpa}}_{\mathsf{tag},i} \leftarrow \mathsf{Enc}^{\mathsf{cpa}}(\mathsf{ek}^{\mathsf{cpa}}_i, x_i, z_i; r_{\mathsf{tag},i})$$

for inputs (x_i, z_i) and randomness $r_{\mathsf{tag},i}$, under the secret key $\mathsf{ek}^{\mathsf{cpa}}_i$, we want to prove that the encryption is well-formed. Almost immediately, one runs into the following main technical issues to resolve:

1. We need to prove well-formedness of encryption using a *simulation-sound* NIZK, where the statement contains $\mathsf{ct}^{\mathsf{cpa}}_{\mathsf{tag},i}$ and the witness includes not only $(x_i, z_i, r_{\mathsf{tag},i})$ but also the secret key $\mathsf{ek}^{\mathsf{cpa}}_i$. The latter involvement of $\mathsf{ek}^{\mathsf{cpa}}_i$ in the witness is crucial to assure that the encryption is well-formed under the secret key, and deviates from the standard NIZK approach for achieving IND-CCA security in PKE. In other words, correct computation of the ciphertext is not enough, as we want to make sure in addition that the encryption key is the coherent one within the system, and not some "wild" key that is not part of the system. In short, what we want to ensure by the NIZK is

   Not only $\mathsf{ct}^{\mathsf{cpa}}_{\mathsf{tag},i}$ *is well-formed under some secret encryption key, but also this secret encryption key is the one that is given to client* i *from setup,* i.e. *from* $\mathsf{Setup}^{\mathsf{cpa}}(1^\lambda, 1^n)$.
2. As we elaborate in the preceding paragraphs, the decryption oracle **Dec** is crucial to capture chosen-ciphertext attacks. In the current context of MCFE, what we will encounter when proving IND-CCA security is the simulation of **Dec**. As a reminder, the classical technique of [63] in achieving IND-CCA security in PKE is to switch between different secret keys within the proof's steps for simulation, while relying on the simulation soundness of the NIZK to assure that no "weird" decryption query that detects the simulation will be asked by the adversary, except with negligible probability. The same technique does not apply to MCFE. First of all *(a)* there is *no fixed decryption key* since the decryption key is associated by a function (which, to make things more complicated, might not be asked to **Extract**). Next, *(b)* the model of MCFE allows the adversary to *corrupt* clients for their secret encryption keys, which means some ciphertext components to **Dec** might be encrypted under corrupted keys, or even under wild encryption keys that do not belong to the system.

We go into each of the issues in the following.

Addressing Issue 1: Well-formedness of Encryption Keys. Our main idea is to use a commitment scheme Com that is *perfectly* binding, and is compatible with the NIZK proof system. Initially, after runninng the setup $\mathsf{Setup}^{\mathsf{cpa}}(1^\lambda, 1^n)$ of $\mathsf{MCFE}^{\mathsf{cpa}}$, all the encryption keys $\mathsf{ek}^{\mathsf{cpa}}_i$ are committed to using the commitment scheme $(c^{\mathsf{cpa}}_{\mathsf{ek},i}, d^{\mathsf{cpa}}_{\mathsf{ek},i}) := \mathsf{Com.Commit}(\mathsf{pp}^{\mathsf{com}}, \mathsf{ek}^{\mathsf{cpa}}_i)$, where $d^{\mathsf{cpa}}_{\mathsf{ek},i}$ is the opening data. The main public parameters of the IND-CCA secure MCFE scheme are then including $(c^{\mathsf{cpa}}_{\mathsf{ek},i})_i$, the IND-CCA encryption keys ek_i include $(\mathsf{ek}^{\mathsf{cpa}}_i, d^{\mathsf{cpa}}_i)$. At the time of encryption, the NIZK uses $(\mathsf{ek}^{\mathsf{cpa}}_i, d^{\mathsf{cpa}}_i)$ as part of the witness to prove the follwing on $c^{\mathsf{cpa}}_{\mathsf{ek},i}$, which becomes part of the statement,

"$c^{\mathsf{cpa}}_{\mathsf{ek},i}$ is a commitment to $\mathsf{ek}^{\mathsf{cpa}}_i$ with opening data $d^{\mathsf{cpa}}_{\mathsf{ek},i}$" AND "the committed value $\mathsf{ek}^{\mathsf{cpa}}_i$ in $c^{\mathsf{cpa}}_{\mathsf{ek},i}$ satisfies the encryption equation $\mathsf{ct}^{\mathsf{cpa}}_{\mathsf{tag},i} = \mathsf{Enc}^{\mathsf{cpa}}(\mathsf{ek}^{\mathsf{cpa}}_i, x_i, z_i; r_{\mathsf{tag},i})$".

At this points it is what we call the *compatibility* between the commitment scheme and the NIZK proof system that allows proving the satisfiability of the committed value (the secret key) with respect to in the encryption equation (fixing the randomness $r_{\mathsf{tag},i}$ gives deterministic calculation of $\mathsf{ct}^{\mathsf{cpa}}_{\mathsf{tag},i}$). The *perfect binding* property intervenes as follows. As a reminder, our current approach depends already on the NIZK to show that

"client i uses some secret key $\widetilde{\mathsf{ek}}^{\mathsf{cpa}}_i$ to encrypt $\mathsf{ct}^{\mathsf{cpa}}_{\mathsf{tag},i} = \mathsf{Enc}^{\mathsf{cpa}}(\widetilde{\mathsf{ek}}^{\mathsf{cpa}}_i, x_i, z_i; r_{\mathsf{tag},i})$".

Integrating with the part of the statement on encryption equation for the comitted value $\mathsf{ek}^{\mathsf{cpa}}_i$, what the NIZK proves (or, its simulation soundness ensures) is the existence of solutions $\widetilde{\mathsf{ek}}^{\mathsf{cpa}}_i$ and $\mathsf{ek}^{\mathsf{cpa}}_i$ that satisfy the encryption equation as the variable for encryption key. When the equation $\mathsf{ct}^{\mathsf{cpa}}_{\mathsf{tag},i} = \mathsf{Enc}^{\mathsf{cpa}}(\mathsf{ek}^{\mathsf{cpa}}_i, x_i, z_i; r_{\mathsf{tag},i})$ has a *unique* solution $\mathsf{ek}^{\mathsf{cpa}}_i$, while fixing variables of $(x_i, z_i, r_{\mathsf{tag},i})$[6], a perfectly binding commitment Com implies that $\widetilde{\mathsf{ek}}^{\mathsf{cpa}}_i = \mathsf{ek}^{\mathsf{cpa}}_i$. This is exactly what we want to show regarding the encryption key: it encrypts to $\mathsf{ct}^{\mathsf{cpa}}_i$ and corresponds to the committed well-formed encryption key from setup. If the commitment is not perfectly binding, a potentially unbounded adversary (we consider NIZK proofs) can prove the satisfiability of the encryption equation with respect to different secret encryption keys $\widetilde{\mathsf{ek}}^{\mathsf{cpa}}_i \neq \mathsf{ek}^{\mathsf{cpa}}_i$, without contradicting the uniqueness of the solution.

So far, the statement of the simulation sound NIZK resembles

"$c^{\mathsf{cpa}}_{\mathsf{ek},i}$ is a Com-commitment to $\mathsf{ek}^{\mathsf{cpa}}_i$ with opening data $d^{\mathsf{cpa}}_{\mathsf{ek},i}$" AND "the committed value $\mathsf{ek}^{\mathsf{cpa}}_i$ in $c^{\mathsf{cpa}}_{\mathsf{ek},i}$ satisfies the encryption equation $\mathsf{ct}^{\mathsf{cpa}}_{\mathsf{tag},i} = \mathsf{Enc}^{\mathsf{cpa}}(\mathsf{ek}^{\mathsf{cpa}}_i, x_i, z_i; r_{\mathsf{tag},i})$" AND "client i uses some secret key $\widetilde{\mathsf{ek}}^{\mathsf{cpa}}_i$ to encrypt $\mathsf{ct}^{\mathsf{cpa}}_{\mathsf{tag},i} = \mathsf{Enc}^{\mathsf{cpa}}(\widetilde{\mathsf{ek}}^{\mathsf{cpa}}_i, x_i, z_i; r_{\mathsf{tag},i})$".

Addressing Issue 2: Simulating Decryption Queries. So as to simulate the **Dec** in the proof for IND-CCA security, we recall that in the formal definition of the original IND-CCA for MCFE, **Dec** works by deriving the functtional key dk_F for a function F that comes with the decryption query, then decrypting the ensemble $(\mathsf{ct}_i)_i$ under dk_F. As the steps in the proof evolves, at some point we find ourselves in an intermediate state where, for instance, we are in the middle of changing the challenge $x^{(b)}_i, z^{(b)}_i$ to $x^{(0)}_i, z^{(0)}_i$. At a high level, one cannot change all occurences of $x^{(b)}_i, z^{(b)}_i$ at once, *e.g.* we have parts of the NIZK statement that touch the commitment Com whereas others relate $\mathsf{MCFE}^{\mathsf{cpa}}$ (so far, see the above NIZK statement). Therefore, because we only restrict the decryption query to not contain pragmatically the challenge ciphertext components $\mathsf{ct}^{(b)}_i$, there are situations where some maliciously crafted **Dec** queries can stem from what we try to change from $x^{(b)}_i, z^{(b)}_i$ to $x^{(0)}_i, z^{(0)}_i$, in particular on the *corrupted components*[7]. This fails the check of the NIZK proof, making decrypting by dk_F in **Dec** under the modified challenge ciphertexts detectable.

Our solution is to use an *equivocable-extractable* (E2-secure) commitment scheme EECom, which is studied in the context of Password-Authenticated Key Exchange or Multi-Party Computation [25, 26, 1] and more. Roughly speaking, an E2-secure commitment scheme comes with an indistinguishable alternative setup algorithm, that gives a trapdoor for simulating commitments that can be equivocated to two different values, while ensuring that one cannot come up with (non-simulated) commitment and opening to a value that is not committed. The properties are strong *as per* [1, Sect. 3.1] in the sense that they hold even when the adversary is given oracle access to extraction oracle and commitment simulation oracle. Having at our disposal an E2-secure commitment scheme EECom, the main idea is to include in the ciphertext ct_i a commitment

$$(c^{\mathsf{ee},j}_{\mathsf{tag},i}, d^{\mathsf{ee}}_{\mathsf{tag},i}) \leftarrow \mathsf{EECom.Commit}(\mathsf{pp}^{\mathsf{ee}}, (x_i, z_i))$$

6 This holds for almost all group-based MCFE for inner products that we consider, thanks to linearity.
7 As we are employing the stronger admissibility from [66, 67], for a corrupted i it can hold that $x^{(b)}_i, z^{(b)}_i \neq x^{(0)}_i, z^{(0)}_i$.

that commits to (x_i, z_i). Later in the proof, we use the alternative setup to get the trapdoor to *(i)* equivocate the *honest* components $(x_i^{(b)}, z_i^{(b)})$ to $(x_i^{(0)}, z_i^{(0)})$, and *(ii)* for *corrupted* client i, to extract (x_i, z_i) from $c_{\mathsf{tag},i}^{\mathsf{ee},j}$ in their decryption query then evaluate directly under the function F to get the decryption result. Finally, we need to add another clause to the NIZK statement to assure that the commitment $c_{\mathsf{tag},i}^{\mathsf{ee},j}$ is opened under the same value (x_i, z_i) that is encrypted in the ciphertext ct_i. To summary, the NIZK statement now includes

> "$c_{\mathsf{ek},i}^{\mathsf{cpa}}$ is a Com-commitment to $\mathsf{ek}_i^{\mathsf{cpa}}$ with opening data $d_{\mathsf{ek},i}^{\mathsf{cpa}}$" AND "the committed value $\mathsf{ek}_i^{\mathsf{cpa}}$ in $c_{\mathsf{ek},i}^{\mathsf{cpa}}$ satisfies the encryption equation $\mathsf{ct}_{\mathsf{tag},i}^{\mathsf{cpa}} = \mathsf{Enc}^{\mathsf{cpa}}(\mathsf{ek}_i^{\mathsf{cpa}}, x_i, z_i; r_{\mathsf{tag},i})$" AND "client i uses some secret key $\widetilde{\mathsf{ek}}_i^{\mathsf{cpa}}$ to encrypt $\mathsf{ct}_{\mathsf{tag},i}^{\mathsf{cpa}} = \mathsf{Enc}^{\mathsf{cpa}}(\widetilde{\mathsf{ek}}_i^{\mathsf{cpa}}, x_i, z_i; r_{\mathsf{tag},i})$" AND "$c_{\mathsf{tag},i}^{\mathsf{ee},j}$ is a EECom-commitment to (x_i, z_i) with opening data $d_{\mathsf{tag},i}^{\mathsf{ee}}$".

Full details of our bootstrapping from CPA to CCA for MCFE are given in Section 4.1. The main security theorem can be found in Theorem 33.

Instantiations for Inner Products. Because we are basing our IND-CCA notion on the strong admissibility condition from [66, 67], we instantiate our generic construction for MCFE with inner products of vectors using the up-to-date scheme from the work [67]. From [67], under SXDH in a bilinear group setting, we start from $\mathsf{MCFE}^{\mathsf{cpa}}[\mathcal{F}_{\mathsf{subvec},B}^{\mathsf{IP}}]$ that is adaptively IND-CPA secure, under static corruption, for the function class $\mathcal{F}_{\mathsf{subvec},B}^{\mathsf{IP}}$. The E2-secure commitment scheme EECom is taken from the work [1], which we recall in Figure 1, and its E2-security holds under SXDH as well. Next, the perfectly-binding commitment Com is taken from the Groth-Sahai-like NIZK family, particularly after enhancing the Groth-Sahai NIZK [49, Sect. 7.1] with techniques from [48, 24, 50, 51] to achieve Groth-Sahai NIZK $\Pi_{\mathsf{gs}}^{\mathsf{ss}}$ with simulation-soundness. The security of the Groth-Sahai NIZK $\Pi_{\mathsf{gs}}^{\mathsf{ss}}$ is based on both the SXDH assumption in the bilinear group setting (from the vanilla Groth-Sahai NIZK [49, Sect. 7.1]) and the DLIN assumption in the prime-order group setting (from the simulation-soundness enhancement techniques [24, 58]). Our final NIZK construction, with respect to statement-witness $(\mathsf{ct}_{\mathsf{tag},i}^{\mathsf{cpa}}, c_i^{\mathsf{ee}}, c_{\mathsf{ek},i}), \quad \omega := (\mathsf{ek}_i^{\mathsf{cpa}}, r_{\mathsf{tag},i}, d_{\mathsf{tag},i}^{\mathsf{ee}}, (x_i, z_i), d_{\mathsf{ek},i})$[8], consists of non black-box integration between *(i)* a Fiat-Shamir transformed $\Pi_{\mathsf{cpa,ee}}^{\mathsf{ss}}$ from 2-special sound Σ-protocol, so as to prove the statement's parts

> "client i uses some secret key $\widetilde{\mathsf{ek}}_i^{\mathsf{cpa}}$ to encrypt $\mathsf{ct}_{\mathsf{tag},i}^{\mathsf{cpa}} = \mathsf{Enc}^{\mathsf{cpa}}(\widetilde{\mathsf{ek}}_i^{\mathsf{cpa}}, x_i, z_i; r_{\mathsf{tag},i})$" AND "$c_{\mathsf{tag},i}^{\mathsf{ee},j}$ is a EECom-commitment to (x_i, z_i) with opening data $d_{\mathsf{tag},i}^{\mathsf{ee}}$"

then the simmulation soundness comes from a result in [40]; And *(ii)* the proof of $\Pi_{\mathsf{gs}}^{\mathsf{ss}}$, which is simulation sound, to prove

> "$c_{\mathsf{ek},i}^{\mathsf{cpa}}$ is a Com-commitment to $\mathsf{ek}_i^{\mathsf{cpa}}$ with opening data $d_{\mathsf{ek},i}^{\mathsf{cpa}}$" AND "the committed value $\mathsf{ek}_i^{\mathsf{cpa}}$ in $c_{\mathsf{ek},i}^{\mathsf{cpa}}$ satisfies the encryption equation $\mathsf{ct}_{\mathsf{tag},i}^{\mathsf{cpa}} = \mathsf{Enc}^{\mathsf{cpa}}(\mathsf{ek}_i^{\mathsf{cpa}}, x_i, z_i; r_{\mathsf{tag},i})$".

Theorem 51 shows that the non-black-box integration of $\Pi_{\mathsf{cpa,ee}}^{\mathsf{ss}}$ and $\Pi_{\mathsf{gs}}^{\mathsf{ss}}$ as above gives a NIZK proof system that satisfies *completeness*, *simulation soundness*, and *zero-knowledge*. Full details can be found in Section 4.2.

MIFE for Inner Products. We now move on to our next main contribution. Our second contribution is placed on MIFE and in the IND-CCA security model, there is now no corruption oracle for the adversary anymore. The reader is encouraged to revisit the high-level reminder of the security experiment at the beginning of this Section 1.2. Putting forward the roadmap towards our constructions for MIFE for$\mathcal{F}_{\mathsf{subvec},B}^{\mathsf{IP}}$: *(i)* We diverge from the trivial implication of the IND-CCA security of MCFE to the IND-CCA security of MIFE, due to the cost of using NIZK as well as heavy mechanism such as E2-secure commitments for the strong primitive

[8] We index all variables by $i \in [n]$ for the ease of following the application of the NIZK in MCFE construction.

of MCFE[9]; *(ii)* We revisit the IND-CCA secure public key IPFE from [18] and observe their induced IND-CCA secure MIFE will be essentially, though their function class is a particular case of our $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$,

$$\mathsf{IPFE} \overset{(1)}{\to} \text{secret-key } \mathsf{IPFE} \overset{(2)}{\to} \mathsf{MI\text{-}IPFE} \overset{(3)}{\to} \mathsf{MI\text{-}IPFE} \text{ w/ } \mathsf{Tag}$$

step (1) privatizes the public key of $\mathsf{MI\text{-}IPFE}$ into some msk for encryption, assuming the keys can be decomposed for encrypting slots i independently[10] step (2) consists of decomposing the msk into multiple encryption keys ek_i, step (3) allows treating tags as simple as relying on the ROM during encryption (for deriving the encryption randomness, for example); *(iii)* It is arguably preferred to aim for a *direct* construction of an IND-CCA secure MIFE that can match the efficiency of the induced MIFE in point *(ii)* while improving what remains from the work [18]:

1. In particular, the IND-CCA secure IPFE scheme from [18] can be generically constructed from the Cramer-Shoup *smooth projective hash function* (SPHF) [34], generalizing the celebrated usage of Cramer-Shoup SPHF to obtain CCA2-security in [34]. However, the main generic construction of [18] for IPFE from SPHF is capped at selective security on the challenge ciphertexts. This remains a technical gap to be filled for the IND-CCA security of MIFE when we follow this path[11].
2. Moreover, by zooming in onto what we can obtain as MIFE from the IND-CCA secure public key IPFE from [18], there is another technical delicacy to address: the IND-CCA security model for MIFE allows repetitions of encryption on fixed (i, tag), and without diving into too much details of the SPHF-based constructions from [18], this means

   for a fixed i, an adversary will be able to observe repetitive statements $\mathbf{b}_i^{(j)}$ that will be hashed by the SPHFs for encryption.

   This is linked to the level of *smoothness* we require from the SPHF. In short, the real encryption uses yes-instance $\mathbf{b}_i^{(j)}$ for encryption, and au course of the proof, we use the hardness of the underlying subset membership problem to switch to no-instance $\widehat{\mathbf{b}_i^{(j)}}$. Under smoothness, hash values under no-instances (conditioned on the projected hash keys) are close to uniformly random values and then hides the challenge bit b. From what we point out above about the repetitive no-instances that can be got by the adversary, the smoothness of the SPHF is then in an exigency to hold for *every* no-instances in the universe of the language, and not only for *fixed random* no-instances, which will not suffice if the adversary deliberately mix-and-matches what they observe as instances. This property is *not* guaranteed by the Cramer-Shoup SPHF, and not in general given the smoothness in the SPHF-based constructions from [18], especially in conjunction with the aim of adaptive security (to address item 1).

We address the above issues in the following. The blueprint of our direct construction for MIFE from SPHF is (1) first building an IND-CPA secure MIFE with adaptive security and with resilience against repetitive no-instances, then (2) upgrading to a *tag-based* IND-CCA secure MIFE, and finally (3) achieving the full IND-CCA security for MIFE using one-time signature and collision-resistant hash functions *as per* the approach in [55]. Both issues 1 and 2 are relevant already in the IND-CPA security of step (1), therefore in the following we focus on the IND-CPA secure MIFE construction to convey our main ideas while referring to Section 5.1 as well as Section 5.2 for the full details of the constructions.

[9] Given our MCFE in Section 4.1 and Section 4.2 concretely for the function class $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$, because our security is guaranteed even against repetitions of encryption on fixed (i, tag), we deduce an MIFE for the same function class while preserving the security level for IND-CCA by fixing a public tag and using that tag for all encryption. This seemingly trivial connection between MCFE and MIFE demonstrates their close relation, and is confirmed in recent works [67, 13].

[10] This holds for many IPFE schemes, *e.g.* the famous DDH-based IPFE from [11] has this property, including the SPHF-based IPFE from [18].

[11] The work of [28] improves to adaptive IND-CCA security while staying public key single client IPFE and using the same blueprint as [18]. We consider this blueprint and aim adaptive security in a different context of MIFE.

Addressing Issue 2: Smoothness of SPHF. The crux of the resolution is a stronger notion of smoothness that suits what we require for the IND-CCA security of MIFE. Beyond the Cramer-Shoup SPHF, we examine a variant by Gennaro-Lindell [43], for which the projection of hash keys is weakened by requiring the instance along with the hash key, but the smoothness is guaranteed computationally for every no-instance in the universe of the language (see [43, Sect. 3]). Using the Gennaro-Lindell SPHF, our generic construction for MIFE from SPHF encrypts tentatively a vector $\mathbf{x}_i$ of length N[12] by computing, using secret encryption key $\mathsf{ek}_i = (K, (\mathsf{pjk}_{i,k})_{k\in[N]}, (\mathsf{hk}_{i,k})_{k\in[N]}, \mathbf{b}, \omega)$,

$$r_i \leftarrow \mathsf{PRF}(K, \mathbf{x}_i); \quad \mathbf{b}_i \leftarrow \mathsf{RSR}(\lambda, \mathbf{b}, r_i); \omega_i \leftarrow \mathsf{RSRw}(\lambda, \mathbf{b}, r_i, \omega); \quad \mathsf{pjk}^{\mathsf{Enc}}_{i,k} \leftarrow \mathsf{projkg}(\mathsf{hk}_{i,k}, \mathbf{b}_i)$$
$$\text{for } k \in [N] : \mathbf{c}_{i,k} \overset{def}{=} \mathsf{projhash}(\mathsf{pjk}^{\mathsf{Enc}}_{i,k}, \mathbf{b}_i, \omega_i) + [\![\mathbf{x}_{i,k}]\!] \in \mathbb{G} \ . \tag{2}$$

where RSR and RSRw are the self-reduction algorithms for the subset membership problem, the SPHF is specified by $\mathsf{PHF} = (\mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$ for the hash keygen, the projection of hash keys, the hash algorithm, and the projected hash algorithm. The i-th slot ciphertext is defined to be $\mathsf{ct}_i \coloneqq (\mathbf{c}_i, \mathbf{b}_i)$ with $\mathbf{c}_i \overset{def}{=} (\mathbf{c}_{i,k})_{k=1}^{N} \in \mathbb{G}^N$ in some cyclic group that contains the hashes of PHF. Then, by including the instance $\mathbf{b}$ in the public parameters, together with appropriate functional key design, the *correctness* can be attained. We require the SPHF PHF to satisfy some CPA-friendliness (see Definition 12, that is first instroduced in [18]), which comes with some form of key-homomorphism property, *i.e.* for all $\mathsf{hk}, \mathsf{hk}' \in \mathcal{K}$ it holds $\mathsf{hash}(\mathsf{hk} + \mathsf{hk}', x) = \mathsf{hash}(\mathsf{hk}, x) + \mathsf{hash}(\mathsf{hk}', x)$, being fundamental for *correctness* of decryption, and more as we will see below.

Concerning IND-CPA security, the Gennaro-Lindell SPHF is smooth enough to protect against mix-and-match on repetitions of instances revealed by encryption on fixed i:

- in the proof we first use hardness of the subset membership problem to switch from yes-instance to no-instance, then properties of CPA-friendliness (notably some *diversity*, in Definition 10, that helps simulating the projection keys and some *translation indistinguishability*, in Definition 11, to translate hash values under no-instances) allows applying the stronger smoothness on the hash of no-instances, then hiding the challenge bit b.
- Although this resolves the problem of mix-and-match given repetitions, using the Gennaro-Lindell SPHF as above is *not enough* to achieve adaptive security for our IND-CPA MIFE. Intuitively, the Gennaro-Lindell SPHF provides a form of proof of membership that is sound only when the yes-instance is fixed before running the projection algorithms. In our context, this selective soundness implies selective security, because otherwise, the IND-CPA adversary can choose adaptively the instances following what is obtained from previous encryption queries.

It is now remaining the question of how to break this selective barrier and achieve adaptive security for our MIFE.

Addressing Issue 1: Adaptive Security from SPHF. In the attempt from Equation (2), there is one layer of Gennaro-Lindell SPHF that is used to encrypt the vector $\mathbf{x}_i$, for each coordinate $\mathbf{x}_{i,k}$. In order to reach adaptive security, a quick glance seems to suggest that we might have to resort to an even stronger notion of SPHF. In particular, the literature on SPHF suggests the one by Katz-Vaikuntanathan [53, 54], which is proven to provide a form of adaptive smoothness, that is, the smoothness holds for every no-instance in the universe of the language, even when the adversary adaptively chooses the no-instances *after seeing the projection hash keys.* Although this will resolve the issue of adaptive security, we will have to verify that the Katz-Vaikuntanathan SPHF is compatible with the CPA-friendliness property that we require for the correctness of decryption. To recall, the Gennaro-Lindell SPHF is essentially a variant of Cramer-Shoup, and

[12] We recall that this extends the function class from what is done in [18], which allows encrypting only scalars at each slot i in the IPFE encryption.

its CPA-friendliness follows similarly what is proved regarding Cramer-Shoup SPHF in [18]. Consequently, we make use of some auxiliary observations to go around the Katz-Vaikuntanathan SPHF:

- Firstly, in our context of IND-CPA security for MIFE, all projection keys are either $(\mathsf{pjk}_{i,k})_k$ generated at setup and included in the secret encryption key ek_i with no corruption, or $(\mathsf{pjk}^{\mathsf{Enc}}_{i,k})_k$ generated at encryption time and used ephemeraly for encryption. The ciphertext components contain no projection keys. Thus, the adversary through the course of interaction with the challenger will not receive any projection keys.
- Secondly, the Gennaro-Lindell SPHF comes with an adavantage of being CPA-friendly, which is a property that is not known to be guaranteed by the Katz-Vaikuntanathan SPHF. More specifically, instead of using one layer of Gennaro-Lindell SPHF, we can use two layers of Gennaro-Lindell SPHF to encrypt the vector $\mathbf{x}_i$. That is, given the encryption key $\mathsf{ek}_i = (K, (\mathsf{pjk}_{i,k})_{k\in[N]}, (\mathsf{hk}_{i,k})_{k\in[N]}, \mathbf{b}, \omega)$, we can encrypt

$$r_i \leftarrow \mathsf{PRF}(K, \mathbf{x}_i); \quad \mathbf{b}_i \leftarrow \mathsf{RSR}(\lambda, \mathbf{b}, r_i); \omega_i \leftarrow \mathsf{RSRw}(\lambda, \mathbf{b}, r_i, \omega); \quad \mathsf{pjk}^{\mathsf{Enc}}_{i,k} \leftarrow \mathsf{projkg}(\mathsf{hk}_{i,k}, \mathbf{b}_i)$$
$$\text{for } k \in [N] : \mathbf{c}_{i,k} \overset{def}{=} \mathsf{projhash}(\mathsf{pjk}^{\mathsf{Enc}}_{i,k}, \mathbf{b}_i, \omega_i) + \mathsf{projhash}(\mathsf{pjk}_{i,k}, \mathbf{b}, \omega) + [\![\mathbf{x}_{i,k}]\!] \in \mathbb{G} \ . \tag{3}$$

  Surprisingly, this two layers idea under the key-homomorphism property of the Gennaro-Lindell SPHF allows to achieve adaptive security for our MIFE. This comes in a similar manner to how [11] uses Cramer-Shoup hash proof system, which can be seen intuitively as two layers of ElGamal, to achieve adaptive security for their IPFE and improving from [5]. In a nutshell, the IND-CPA proof now contains a step that switches the challenge $\left[\!\left[\mathbf{x}^{(b,j)}_{i,k}\right]\!\right]$ (up to j-repetitions) to $\left[\!\left[\mathbf{x}^{(0,j)}_{i,k}\right]\!\right]$, by modifying the first $\mathbf{b}_i$-dependent SPHF layer $\mathsf{projhash}(\mathsf{pjk}^{\mathsf{Enc}}_{i,k}, \mathbf{b}_i, \omega_i)$. At the same time, the second $\mathbf{b}_i$-independent SPHF layer $\mathsf{projhash}(\mathsf{pjk}_{i,k}, \mathbf{b}, \omega)$ accumulates the differences that result from the switch, and keeps the changes indistinguishable thanks to key-homomorphism on $(\mathbf{b}, \omega)$ as well as the *admissibility* of inner products for MIFE on $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$ (that is, $\sum_i \langle \mathbf{x}^{(0,j)}_i - \mathbf{x}^{(1,j)}_i, \mathbf{y}_i\rangle = 0$ over all i in the sum as we do not have corruption). Last but not least, in the actual proof it will be the hash values that we modify, since the instances are changed prior to no-instances (for smoothness).

We refer to the transition $\mathsf{G}_4 \to \mathsf{G}_5$ in the proof of Theorem 52 for full details of this step. This then completes the construction of an IND-CPA secure MIFE for inner products, with adaptive security and resilience against repetitions of encryption on fixed i. Our generic SPHF-based construction, up to the tag-based IND-CCA secure MIFE, is detailed in Section 5.1 and achieves the same total asymptotic efficiency as the public-key IND-CCA secure IPFE from [18] (see Remark 40), with stronger adaptive security and richer function class $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$ in a totally different context of MIFE. In Section 5.2 we provide instantiations for our MIFE from the DDH and DCR assumptions.

2 Preliminaries

We write $[n]$ to denote the set $\{1, 2, \ldots, n\}$ for an integer n. For any $q \geq 2$, we let $\mathbb{Z}_q$ denote the ring of integers with addition and multiplication modulo q. For a prime q and an integer N, we denote by $GL_N(\mathbb{Z}_q)$ the general linear group of of degree N over $\mathbb{Z}_q$. We write vectors as row-vectors, unless stated otherwise. For a vector $\mathbf{x}$ of dimension n, the notation $\mathbf{x}[i]$ indicates the i-th coordinate of $\mathbf{x}$, for $i \in [n]$. We will follow the implicit notation in [39] and use $[\![a]\!]$ to denote g^a in a cyclic group $\mathbb{G}$ of prime order q generated by g, given $a \in \mathbb{Z}_q$. This implicit notation extends to matrices and vectors having entries in $\mathbb{Z}_q$. We use the shorthand ppt for "probabilistic polynomial time".

Hardness Assumptions. We need the **Decisional Diffie-Hellman** (DDH) assumption in a cyclic group $\mathbb{G}$ of prime order q. In a cyclic group $\mathbb{G}$ of prime order q, the (DDH) assumption in $\mathbb{G}$ assumes that no ppt adversary can distinguish the distributions $\{([\![1]\!], [\![a]\!], [\![b]\!], [\![ab]\!])\}$ and $\{([\![1]\!], [\![a]\!], [\![b]\!], [\![c]\!])\}$ for $a, b, c \xleftarrow{\$} \mathbb{Z}_q$. In the bilinear setting $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, \mathbf{e}, q)$, the **Symmetric eXternal Diffie-Hellman** (SXDH) assumption makes the DDH assumption in both $\mathbb{G}_1$ and $\mathbb{G}_2$. We also use the **Decision Linear assumption** (DLIN) in a cyclic group $\mathbb{G} = \langle g \rangle$, which assumes that no ppt adversary can distinguish the distributions $\{([\![1]\!], [\![a]\!], [\![b]\!], [\![ac]\!], [\![bd]\!], [\![c+d]\!])\}$ and $\{([\![1]\!], [\![a]\!], [\![b]\!], [\![ac]\!], [\![bd]\!], [\![z]\!])\}$ for $a, b, c, d, z \xleftarrow{\$} \mathbb{Z}_q$. We also need the **Decisional Composite Residuosity** (DCR) assumption. Let a composite $N = pq$, for primes p, q, and let an integer $\zeta \geq 1$. The **ζ-Decision Composite Residuosity** (ζ-DCR) problem is to distinguish between the distributions $D_0 := \{z = z_0^{N^\zeta} \bmod N^{\zeta+1} \mid z_0 \xleftarrow{\$} U(\mathbb{Z}_N^*)\}$ and $D_1 := \{z \xleftarrow{\$} U(\mathbb{Z}_{N^{\zeta+1}}^*)\}$. For each $\zeta > 0$, the ζ-DCR assumption was shown to be equivalent to the original 1-DCR assumption [36]. However, the semantic security of their scheme under the 1-DCR assumption for any polynomial ζ is a well-known result. The proof of Lemma 1 is perhaps folklore, for instance, a full proof can be found in [38].

Lemma 1. *Let $\zeta = \mathsf{poly}(\lambda)$. Then ζ-DCR is equivalent to 1-DCR with a security loss at most ζ.*

2.1 Commitment Scheme

A *commitment scheme* is a tuple of PPT algorithms $\mathsf{CI} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify})$ such that

- $\mathsf{Setup}(1^\lambda)$: generates the public parameters pp,
- $\mathsf{Commit}(\mathsf{pp}, m)$: given the public parameters pp, message $m \in \mathcal{C}_{\mathsf{msg}}$, computes a commitment $c \in \mathcal{C}_{\mathsf{com}}$ with opening randomness d, and outputs the pair (c, d),
- $\mathsf{Verify}(\mathsf{pp}, c, m, d)$: given the public parameters pp, message $m \in \mathcal{C}_{\mathsf{msg}}$, and opening randomness d, outputs a bit $b \in \{0, 1\}$ which depends on the validity of the opening (m, d) with respect to the commitment c.

Here, $\mathcal{C}_{\mathsf{msg}}$, $\mathcal{C}_{\mathsf{rnd}}$, $\mathcal{C}_{\mathsf{com}}$, are message, randomness, and commitment spaces, respectively. If the public parameters are uniform or explainable (*i.e.*, Setup outputs some $\mathsf{pp} \xleftarrow{\$} \{0,1\}^\ell$ for $\ell \in \mathbb{N}$) we omit Setup without loss of generality.

We require the correctness, hiding and binding properties for a commitment scheme. A commitment schemes is *correct*, if honest commitments $(c, d) \leftarrow \mathsf{Commit}(\mathsf{pp}, m; r)$ always verify, *i.e.* it holds that $\mathsf{Verify}(\mathsf{pp}, c, m, d) = 1$ where pp are the public parameters. It is *hiding* if it is hard to decide whether an unopened commitment c commits to message m_0 or m_1, and it is *binding* if it is hard to open commitments c to distinct messages. We can have *computational, statistical, perfect* variants for hiding and binding properties.

Perfectly Binding Commitment Scheme. In order to get a perfectly binding commitment scheme, together with the hiding property, one classical approach is using a public key encryption (PKE) scheme: the encryption algorithm will be the commitment procedure, the randomness used in the encryption will be the opening data, and using the decryption key we can even extract the message from the commitment. Let us describe the commitment scheme $\mathsf{Com} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify})$ that is based on the ElGamal PKE scheme:

– $\mathsf{Setup}(1^\lambda)$: sets up a group $\mathbb{G} = \langle g \rangle$ of prime order q and generates a generator g, chooses a random $x \xleftarrow{\$} \mathbb{Z}_q$ and sets the CRS $\mathsf{crs} = (\mathbb{G}, g, h = g^x)$,
– $\mathsf{Commit}(\mathsf{crs}, m)$: given the CRS $\mathsf{crs} = (\mathbb{G}, g, h)$ and a message $m \in \mathbb{G}$, chooses random $r \xleftarrow{\$} \mathbb{Z}_q$, computes the commitment $c = (g^r, m \cdot h^r)$, as well as its opening $d = r$,
– $\mathsf{Verify}(\mathsf{crs}, c, m, d)$: given the CRS $\mathsf{crs} = (\mathbb{G}, g, h)$, a commitment $c = (c_1, c_2)$, a message $m \in \mathbb{G}$, and an opening d, outputs 1 if $c = (g^d, m \cdot h^d)$ and 0 otherwise.

The above commitment scheme is computationally hiding under DDH following the $\mathsf{ind\text{-}cpa}$ security of the ElGamal PKE scheme, and is perfectly binding due to the perfect correctness of the ElGamal PKE scheme. The Cramer-Shoup PKE scheme [34] yields a commiment scheme that is computationally hiding, perfectly binding, with non-malleability, and extractable. We recall the details below and use it (for the equivocable and extractable commitment) in our CCA-secure MCFE construction in Section 4.1.

Equivocable and Extractable (E2-secure) Commitment Scheme. We will use a more advanced commitment scheme that is equivocable and extractable, which is a result from [1]. We use a variant of the Cramer-Shoup PKE scheme, in order to deal with vectors. The setting is a cyclic group $\mathbb{G} = \langle g \rangle = \langle h \rangle$ of prime order q, where the two generators g and h are independent. We write $\mathbb{G}$ additively for the coherence of notations with our additive notation in bilinear group settings. The secret decryption key is $\mathsf{sk} = (a, b, c, d, e) \xleftarrow{\$} \mathbb{Z}_q^5$, while the public key is $\mathsf{pk} = (g, h, U = a \cdot g + b \cdot h, V = c \cdot g + d \cdot h, W = e \cdot g, \mathsf{H}^{\mathsf{cr}})$. The function H^{cr} is randomly chosen in a collision-resistant hash function family $\mathcal{H}$. For a vector $M = (M_i)_{i \in [N]} \in \mathbb{G}^n$, the encryption of M by multi-Cramer-Shoup is given by

$$N\text{-}\mathsf{MCS}^{\mathsf{tag}}(\mathsf{pk}, M; (\mathsf{rnd}_i)_i) \stackrel{def}{=} (u_i = \mathsf{rnd}_i \cdot g, v_i = \mathsf{rnd}_i \cdot h, \\ t_i = M_i + \mathsf{rnd}_i \cdot W, w_i = \mathsf{rnd}_i (U + \sigma V)) \in \mathbb{G}^4$$

where $\sigma = \mathsf{H}^{\mathsf{cr}}(\mathsf{tag}, (u_i, v_i, t_i)_i)$[13]. The decryption of ciphertexts $(u_i, v_i, t_i, w_i)_i$ is done by $M_i = t_i - e \cdot u_i$ after checking $w_i \stackrel{?}{=} (a + c\sigma) \cdot u_i + (b + d\sigma) \cdot v_i$ for each $i \in [N]$. Under the DDH assumption, the multi-Cramer-Shoup encryption $N\text{-}\mathsf{MCS}^{\mathsf{tag}}$ is IND-CCA secure.

Based on multi-Cramer-Shoup, the equivocable-extractable commitment scheme $\mathsf{EECom} = (\mathsf{Setup}, \mathsf{SetupTr}, \mathsf{Commit}, \mathsf{ExtCom}, \mathsf{OpenCom}, \mathsf{SimCom}, \mathsf{Verify})$ from [1] is described in Figure 1.

Definition 2 (E2-Security of Commitments, [1]). *A commitment scheme* $\mathsf{EECom} = (\mathsf{Setup}, \mathsf{SetupTr}, \mathsf{Commit}, \mathsf{ExtCom}, \mathsf{OpenCom}, \mathsf{SimCom}, \mathsf{Verify})$ *is* equivocable-extractable *if it satisfies the following properties:*

Strong Simulation Indistinguishability: *one cannot distinguish a real commitment (by* Commit*) from a simulated one (by* SimCom*), even with oracle access to the extraction oracle* ExtCom *and with oracle access to simulation oracle* SimCom*.*

Strong Binding Extractability: *one cannot produce a commitment and a valid opening (that do not come from the simulated procedure* SimCom*) to an input x while the commitment is not committing to x, even with oracle access to the extraction oracle* ExtCom *and with oracle access to simulation oracle* SimCom*.*

2.2 Smooth Projective Hash Functions (SPHF) with FE-Friendliness

We recall the definition of *smooth projective hash functions*, which was first introduced in [35] and a variant was studied in [43]. Our paper will use the syntactical variant *as per* [43, Section 3].

Subset membership problems. A subset membership problem $\mathbf{P}$ is a collection $\{I_\lambda\}_{\lambda \in \mathbb{N}}$, each I_λ is a distribution over *problem instances* Λ that specifies

- Finite, non-empty sets $\mathcal{X}_\lambda, \mathcal{U}_\lambda \subseteq \{0,1\}^{poly(\lambda)}$ and distributions $D(\mathcal{X}_\lambda), D(\mathcal{U}_\lambda \setminus \mathcal{X}_\lambda)$ over $\mathcal{X}_\lambda, \mathcal{U}_\lambda \setminus \mathcal{X}_\lambda$ respectively.
- A witness set $\mathcal{Y}_\lambda \subseteq \{0,1\}^{poly(\lambda)}$ and an NP-relation $\mathsf{Rel}_\lambda \subseteq \mathcal{U}_\lambda \times \mathcal{Y}_\lambda$ that satisfies: for any $x \in \mathcal{X}_\lambda$, there exists $y \in \mathcal{Y}_\lambda$ such that $(x, y) \in \mathsf{Rel}_\lambda$.

[13] The hash value σ is the same for all $i \in [N]$.

$\mathsf{SetupTr}(1^n)$ sets up the bilinear setting $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, \mathbf{e}, q)$, with another generator h_1 of $\mathbb{G}_1$. The public key for multi-Cramer-Shoup is $\mathsf{pk} = (g, h, U \overset{def}{=} a \cdot g_1 + b \cdot h_1, V \overset{def}{=} c \cdot g_1 + d \cdot h_1, W \overset{def}{=} e \cdot g_1, \mathsf{H}^{\mathsf{cr}})$, where $a, b, c, d, e \overset{\$}{\leftarrow} \mathbb{Z}_q$. The hash function H^{cr} is chosen from a collision-resistant hash function family $\mathcal{H}$. A random $t \overset{\$}{\leftarrow} \mathbb{Z}_q$ is chosen and the CRS is $\mathsf{crs} = (\mathsf{pk}, T = t \cdot g_2)$, while the trapdoor is $\mathsf{td} = (a, b, c, d, e, t)$, *i.e.* the secret key together with t. The non-trapdoor set up Setup is the same as $\mathsf{SetupTr}$, but no trapdoor is output.

$\mathsf{Commit}(\mathsf{crs}, \mathsf{tag}, M)$ takes the CRS $\mathsf{crs} = (\mathsf{pk}, T)$, a tag tag, and commits to a vector $M \in \{0,1\}^N$. The commitment works as follows:

- For $i \in [N]$, it chooses random $\mathsf{rnd}_{i,M_i} \overset{\$}{\leftarrow} \mathbb{Z}_q$ and sets $\mathsf{rnd}_{i,1-M_i} = 0$. Using rnd_{i,M_i} as randomness, it computes

$$a_i = \left[\!\left[\mathsf{rnd}_{i,M_i}\right]\!\right]_2 + M_i \cdot T; \quad \forall j \in \{0,1\} : \ d_{i,j} = [\![\mathsf{rnd}_{i,j}]\!]_1 .$$

  We denote $\mathbf{a} = (a_i)_{i \in [N]}$.
- For $i \in [N]$ and $j \in \{0,1\}$, it uns the multi-Cramer-Shoup on length $2N$ with $(\mathsf{tag}, \mathbf{a})$ to compute

$$\mathbf{b} = (b_{i,j})_{i \in [N], j \in \{0,1\}} = 2N\text{-}\mathsf{MCS}^{(\mathsf{tag},\mathbf{a})}(\mathsf{pk}, \mathbf{d}; \mathbf{s}) = (u_{i,j}, v_{i,j}, t_{i,j}, w_{i,j})_{i \in [N], j \in \{0,1\}}.$$

  The randomness $\mathbf{s} = (s_{i,j})_{i \in [N], j \in \{0,1\}} \overset{\$}{\leftarrow} \mathbb{Z}_q$ is used in the multi-Cramer-Shoup encryption, the vector $\mathbf{d} = (d_{i,j})_{i,j}$ is computed above.

The commitment is $C = (\mathbf{a}, \mathbf{b})$, the opening is $D = (s_{i,M_i})_{i \in [N]}$.

$\mathsf{ExtCom}(\mathsf{crs}, \mathsf{tag}, \mathsf{td}, C)$ uses the trapdoor td to decrypt $\mathbf{b}$ from the commitment C to obtain $(d_{i,j})_{i,j}$. It then checks $e(g_1, a_{i,j} - j \cdot T) \overset{?}{=} e(d_{i,j}, g_2)$ for all $i \in [N]$ and $j \in \{0,1\}$. If for all $i \in [N]$, there is excatly one $j = M_i \in \{0,1\}$ making the check pass, output M_i as the i-th bit of the message M. Else, it outputs $\perp$.

$\mathsf{OpenCom}(\mathsf{crs}, \mathsf{tag}, \mathsf{eqk}, C, M)$ takes the equivocate $\mathsf{eqk} = \mathbf{s}$ to make the openning $D = (s_{i,M_i})_{i \in [N]}$ so as to open to the message M.

$\mathsf{SimCom}(\mathsf{crs}, \mathsf{tag}, \mathsf{td})$ tkaes the trapdoor td, using t to output $C = (\mathbf{a}, \mathbf{b})$ along with an equivocate $\mathsf{eqk} = \mathbf{s}$ such that

- For $i \in [N]$, it chooses random $\mathsf{rnd}_{i,0} \overset{\$}{\leftarrow} \mathbb{Z}_q$ and sets $\mathsf{rnd}_{i,1} = \mathsf{rnd}_{i,0} - t$. It then commits to both 0 and 1 using $\mathsf{rnd}_{i,0}$ and $\mathsf{rnd}_{i,1}$

$$a_i = [\![\mathsf{rnd}_{i,0}]\!]_2 = [\![\mathsf{rnd}_{i,1}]\!]_2 + T; \quad \forall j \in \{0,1\} : \ d_{i,j} = [\![\mathsf{rnd}_{i,j}]\!]_1 .$$

  This makes $d_{i,j}$ the opening for a_i to $j \in \{0,1\}$. The vector $\mathbf{a} = (a_i)_{i \in [N]}$.
- the vector $\mathbf{b}$ is computed as in Commit.

$\mathsf{Verify}(\mathsf{crs}, \mathsf{tag}, C, M, D)$ takes $C = (\mathbf{a}, \mathbf{b})$, checks the validity of the ciphertext b_{i,M_i} in $\mathbf{b}$ using s_{i,M_i} in D. Knowing the randomness of the multi-Cramer-Shoup encryption, it also extracts d_{i,M_i} from b_{i,M_i} and s_{i,M_i}. It then checks $e(g_1, a_{i,M_i} - M_i \cdot T) \overset{?}{=} e(d_{i,M_i}, g_2)$ for all $i \in [N]$.

Fig. 1: The equivocable-extractable commitment scheme EECom of [1].

A subset membership problem is *efficiently sampleable* if the following ppt algorithms exist

– (Problem instance sampleability) Given 1^λ, sample an instance $\Lambda \overset{\$}{\leftarrow} I_\lambda$, specifying $\Lambda = (\mathcal{U}_\lambda, \mathcal{X}_\lambda, D(\mathcal{X}_\lambda), D(\mathcal{U}_\lambda \setminus \mathcal{X}_\lambda), \mathcal{Y}_\lambda, \mathsf{Rel}_\lambda)$.
– (Instance member sampleability) Given 1^λ and a problem instance Λ, sample $x \overset{\$}{\leftarrow} D(\mathcal{X}_\lambda)$ together with a witness $y \in \mathcal{Y}_\lambda$ such that $(x, y) \in \mathsf{Rel}_\lambda$.
– (Instance non-member sampleability) Given 1^λ and a problem instance Λ, sample $x \overset{\$}{\leftarrow} D(\mathcal{U}_\lambda \setminus \mathcal{X}_\lambda)$.

Definition 3. *Let $\mathbf{P} = \{I_\lambda\}_{\lambda \in \mathbb{N}}$ be a subset membership problem. For each distribution $I_\lambda \in \mathbf{P}$, we define $\mathsf{dom}(I_\lambda)$ to be its support. We define the random variables*

1. *$V(\mathcal{X}_\lambda)$ over $\mathsf{dom}(I_\lambda) \times \{0,1\}^{poly(\lambda)}$ that is obtained by sampling a problem instance $\Lambda \overset{\$}{\leftarrow} I_\lambda$, then $x \overset{\$}{\leftarrow} D(\mathcal{X}_\lambda)$ from the specification of Λ, and outputing (Λ, x).*
2. *$V(\mathcal{U}_\lambda \setminus \mathcal{X}_\lambda)$ over $\mathsf{dom}(I_\lambda) \times \{0,1\}^{poly(\lambda)}$ that is obtained by sampling a problem instance $\Lambda \overset{\$}{\leftarrow} I_\lambda$, then $x \overset{\$}{\leftarrow} D(\mathcal{U}_\lambda \setminus \mathcal{X}_\lambda)$ from the specification of Λ, and outputing (Λ, x).*

The subset membership problem $\mathbf{P}$ is said to be hard *if*

$$D_0 = \{V(\mathcal{X}_\lambda)\}_{\lambda \in \mathbb{N}} \qquad\qquad D_1 = \{V(\mathcal{U}_\lambda \setminus \mathcal{X}_\lambda)\}_{\lambda \in \mathbb{N}}$$

are computationally indistinguishable*, where the probability is taken over the coins of all involved* ppt *algorithms.*

Average-case Random Self-reducibility. In particular, the (induced relation of) subset memebership problems for SPHFs are required to be *average-case random-self reducible*, of which the notion was introduced in [80] and is recalled in Definition 4.

Definition 4 (Average self-reducibility [80]). *Let $\lambda \in \mathbb{N}$. We consider families of finite sets $\mathcal{X} = \{\mathcal{X}_\lambda\}_\lambda$, $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_\lambda$, $\mathcal{X}' = \{\mathcal{X}'_\lambda\}_\lambda$, $\mathcal{Y}' = \{\mathcal{Y}'_\lambda\}_\lambda$. We define relations $\mathsf{Rel}_\lambda \subseteq \mathcal{X}_\lambda \times \mathcal{Y}_\lambda$ and $\mathsf{Rel}'_\lambda \subseteq \mathcal{X}'_\lambda \times \mathcal{Y}'_\lambda$. Let*

$$\mathsf{dom}(\mathsf{Rel}_\lambda) := \{x \in \mathcal{X}_\lambda : \exists y \in \mathcal{Y}_\lambda \wedge (x, y) \in \mathsf{Rel}_\lambda\} \ ,$$

similarly

$$\mathsf{dom}(\mathsf{Rel}'_\lambda) := \{x \in \mathcal{X}'_\lambda : \exists y \in \mathcal{Y}'_\lambda \wedge (x, y) \in \mathsf{Rel}'_\lambda\} \ .$$

For any $x \in \mathcal{X}_\lambda, x' \in \mathcal{X}'_\lambda$, we write $\mathsf{Rel}_\lambda(x) := \{y : (x, y) \in \mathsf{Rel}_\lambda\}$ and $\mathsf{Rel}'_\lambda(x') := \{y' : (x', y') \in \mathsf{Rel}'_\lambda\}$. The relation

$$\mathcal{R} := \{((\lambda, x), y) : \lambda \in \mathbb{N} \wedge (x, y) \in \mathsf{Rel}_\lambda\}$$

is average reducible *to the relation*

$$\mathcal{R}' := \{((\lambda, x'), y') : \lambda \in \mathbb{N} \wedge (x', y') \in \mathsf{Rel}'_\lambda\}$$

iff there exists a ppt *algorithm* RSR *such that*

1. *Given as inputs $(\lambda \in \mathbb{N}, x \in \mathsf{dom}(\mathsf{Rel}_\lambda), r \overset{\$}{\leftarrow} \{0,1\}^{poly(\lambda)})$, A outputs $x' \in \mathsf{dom}(\mathsf{Rel}'_\lambda)$. Moreover x' is uniformly distributed over $\mathsf{dom}(\mathsf{Rel}'_\lambda)$.*
2. *There exists a* ppt *algorithm* RSRw *that satisfies*

$$\mathsf{RSRw}(\lambda, x, r, y) \rightarrow y' \in \mathsf{Rel}'_\lambda(\mathsf{RSR}(\lambda, x, r))$$

   given as inputs $\lambda \in \mathbb{N}, x \in \mathsf{dom}(\mathsf{Rel}_\lambda), r \overset{\$}{\leftarrow} \{0,1\}^{poly(\lambda)}$ and $y \in \mathsf{Rel}_\lambda(x)$.
3. *There exists a* ppt *algorithm* RSRw′ *that satisfies*

$$\mathsf{RSRw}'(\lambda, x, r, y') \rightarrow y \in \mathsf{Rel}_\lambda(x)$$

   given $\lambda \in \mathbb{N}, x \in \mathsf{dom}(\mathsf{Rel}_\lambda), r \overset{\$}{\leftarrow} \{0,1\}^{poly(\lambda)}$ and $y' \in \mathsf{Rel}_\lambda(\mathsf{RSR}(\lambda, x, r))$.

The relation Rel_λ is average self-reducible *if Rel_λ is average reducible to itself.*

Smooth projective hash functions. Let **P** be a subset membership problem that is efficiently sampleable.

Definition 5 ([43]). *A SPHF PHF = (hashkg, projkg, hash, projhash) is recalled below*

hashkg(Λ): *Given as input a problem instance $\Lambda \overset{\$}{\leftarrow} I_\lambda$ and output a key $\mathsf{hk} \in \mathcal{K}$.*
projkg(hk, x): *Given a hash key $\mathsf{hk} \in \mathcal{K}$ $x \in \mathcal{X}_\lambda$ with respect to $\Lambda = (\mathcal{U}_\lambda, \mathcal{X}_\lambda, D(\mathcal{X}_\lambda), D(\mathcal{U}_\lambda \setminus \mathcal{X}_\lambda), \mathcal{Y}_\lambda, \mathsf{Rel}_\lambda)$, output a projection key* pjk.
hash(hk, x): *Given as inputs a hahs key $\mathsf{hk} \in \mathcal{K}$ and $x \in \mathcal{U}_\lambda$ with respect to $\Lambda = (\mathcal{U}_\lambda, \mathcal{X}_\lambda, D(\mathcal{X}_\lambda), D(\mathcal{U}_\lambda \setminus \mathcal{X}_\lambda), \mathcal{Y}_\lambda, \mathsf{Rel}_\lambda)$, output a hash value* hval.
projhash(pjk, x, y): *Given a projection key* pjk *and $x \in \mathcal{X}_\lambda$, $y \in \{0,1\}^{poly(\lambda)}$, output a projected hash value* pjhval.

Definition 6 (Completeness). *A SPHF PHF = (hashkg, projkg, hash, projhash) is* complete *if for all $\lambda \in \mathbb{N}$, all $\Lambda \overset{\$}{\leftarrow} I_\lambda$ specifying $\Lambda = (\mathcal{U}_\lambda, \mathcal{X}_\lambda, D(\mathcal{X}_\lambda), D(\mathcal{U}_\lambda \setminus \mathcal{X}_\lambda), \mathcal{Y}_\lambda, \mathsf{Rel}_\lambda)$, all $\mathsf{hk} \leftarrow \mathsf{hashkg}(\Lambda)$, all $x \in \mathcal{X}_\lambda$ and $y \in \mathcal{Y}_\lambda$ such that $(x, y) \in \mathsf{Rel}_\lambda$, if $\mathsf{pjk} \leftarrow \mathsf{projkg}(\mathsf{hk}, x)$, then*

$$\mathsf{hash}(\mathsf{hk}, x) = \mathsf{projhash}(\mathsf{pjk}, x, y) \ .$$

Definition 7 (Key-homomorphism). *A SPHF PHF = (hashkg, projkg, hash, projhash) is* key-homomorphic *if*

1. *The key space $\mathcal{K}$ of hash keys and range $\mathcal{R}$ of hash values are Abelian groups with* ppt *group operations.*
2. *For all $\lambda \in \mathbb{N}$, all $\Lambda \overset{\$}{\leftarrow} I_\lambda$ specifying $\Lambda = (\mathcal{U}_\lambda, \mathcal{X}_\lambda, D(\mathcal{X}_\lambda), D(\mathcal{U}_\lambda \setminus \mathcal{X}_\lambda), \mathcal{Y}_\lambda, \mathsf{Rel}_\lambda)$, all $\mathsf{hk} \leftarrow \mathsf{hashkg}(\Lambda)$, we have: for all $\mathsf{hk}, \mathsf{hk}' \in \mathcal{K}$*

$$\mathsf{hash}(\mathsf{hk} + \mathsf{hk}', x) = \mathsf{hash}(\mathsf{hk}, x) + \mathsf{hash}(\mathsf{hk}', x) \ .$$

```
$\mathsf{Expr}^{\mathbf{P},\mathcal{A}}(1^\lambda)$:

  $b \xleftarrow{\$} U(\{0,1\})$
  $\Lambda \xleftarrow{\$} I_\lambda$
  $b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{hash}}(\cdot),\, \mathcal{O}_{\mathcal{X}}}(\Lambda)$
  Output $b' \overset{?}{=} b$
```

where the oracles are defined as: $\mathsf{dom}(\mathcal{O}_{\mathcal{X}}) := \varnothing$
$\Lambda := (\mathcal{U}_\lambda, \mathcal{X}_\lambda, D(\mathcal{X}_\lambda), D(\mathcal{U}_\lambda \setminus \mathcal{X}_\lambda), \mathcal{Y}_\lambda, \mathsf{Rel}_\lambda)$;

$$\mathcal{O}_{\mathsf{hash}}^{\Lambda,\mathcal{K},\mathcal{R}}(x) : \begin{cases} \text{sample } \mathsf{hk} \xleftarrow{\$} \mathcal{K}; \mathsf{hval} \xleftarrow{\$} \mathcal{R} \\ \text{output } (\mathsf{projkg}(\mathsf{hk}, x), \mathsf{hash}(\mathsf{hk}, x)) \\ \qquad \text{if } b = 1 \wedge x \in \mathsf{dom}(\mathcal{O}_{\mathcal{X}}) \\ \text{output } \perp \\ \qquad \text{if } b = 1 \wedge x \notin \mathsf{dom}(\mathcal{O}_{\mathcal{X}}) \\ \text{output } (\mathsf{projkg}(\mathsf{hk}, x), \mathsf{hval}) \\ \qquad \text{if } b = 0 \end{cases}$$

$$\mathcal{O}_{\mathcal{X}}^{\Lambda} : \begin{cases} x \xleftarrow{\$} D(\mathcal{X}_\lambda); \mathsf{dom}(\mathcal{O}_{\mathcal{X}}) := \mathsf{dom}(\mathcal{O}_{\mathcal{X}}) \cup \{x\} \\ \text{output } x \end{cases} .$$

Fig. 2: Computational smoothness for yes-instances with respect to the subset problem $\mathbf{P}$.

Smoothness. We define *smoothness* following the variant in [43, Section 3]. In the later section on CCA-security of MIFE, we will need the computational smoothness with respect to the no-instance.

Definition 8 (Smoothness). *Let $\mathbf{P} = \{I_\lambda\}_{\lambda\in\mathbb{N}}$ be a subset membership problem. For each distribution $I_\lambda \in \mathbf{P}$, we define $\mathsf{dom}(I_\lambda)$ to be its support. We define the random variables, sampling $\Lambda \xleftarrow{\$} I_\lambda$ specifying $\Lambda = (\mathcal{U}_\lambda, \mathcal{X}_\lambda, D(\mathcal{X}_\lambda), D(\mathcal{U}_\lambda \setminus \mathcal{X}_\lambda), \mathcal{Y}_\lambda, \mathsf{Rel}_\lambda)$ and fixing $x \in \mathcal{X}_\lambda$,*

1. *$V(x, \mathsf{projkg}(\mathsf{hk}, x), \mathsf{hash}(\mathsf{hk}, x))$ that is obtained by generating $\mathsf{hk} \leftarrow \mathsf{hashkg}(\Lambda)$, and outputing $(x, \mathsf{projkg}(\mathsf{hk}, x), \mathsf{hash}(\mathsf{hk}, x))$.*
2. *$V(x, \mathsf{projkg}(\mathsf{hk}, x), \mathsf{hval})$ that is obtained by generating $\mathsf{hk} \leftarrow \mathsf{hashkg}(\Lambda)$, sampling $\mathsf{hval} \xleftarrow{\$} \mathcal{R}_\lambda$ and outputing $(x, \mathsf{projkg}(\mathsf{hk}, x), \mathsf{hval})$.*

The SPHF PHF is said to be smooth *if for all $x \in \mathcal{U}_\lambda \setminus \mathcal{X}_\lambda$*

$$D_0 = \{V(x, \mathsf{projkg}(\mathsf{hk}, x), \mathsf{hash}(\mathsf{hk}, x))\}_{\lambda\in\mathbb{N}} \qquad D_1 = \{V(x, \mathsf{projkg}(\mathsf{hk}, x), \mathsf{hval})\}_{\lambda\in\mathbb{N}}$$

are statistically indistinguishable*, where the probability is taken over the coins of all involved* ppt *algorithms.*

Computational smoothness of yes-instance. We need the following technical lemma form [43, Lemma 3.1].

Lemma 9 ([43]). *Let $\mathbf{P}$ be a* hard *subset membership problem. Then, for sufficiently large $\lambda \in \mathbb{N}$, the following difference*

$$\mathsf{Adv}^{\mathbf{P},\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Expr}^{\mathbf{P},\mathcal{A}}(1^\lambda) = 1\right] - 1/2\right|$$

is negligible in λ, where $\mathsf{Expr}^{\mathbf{P},\mathcal{A}}(1^\lambda)$ is described in Figure 2.

CPA-Friendliness. We recall the definition of CPA-friendliness of SPHF from [18]. The required properties are summarized in the following definitions.

Definition 10 (Strong Diversity). *A* key-homomorphic *SPHF*

$$\mathit{PHF} = (\mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$$

for a subset membership problem $\mathbf{P}$, with yes-instances in $\mathcal{X}$ and no-instances in $\bar{\mathcal{X}} \overset{def}{=} \mathcal{U} \setminus \mathcal{X}$ with respect to a universe $\mathcal{U}$, is $(\mathsf{hk}_\perp, g_\perp, M_\perp)$-strongly diverse for

- *A function $\mathsf{hk}_\perp : \bar{\mathcal{X}} \to \mathbb{G}$,*
- *a group element $g_\perp \in \mathbb{G}$,*

- *a positive natural number* $M_\perp \in \mathbb{N}_{>0}$

if the following holds

1. $g_\perp, M_\perp$ *can be computed in polynomial time from the problem instance* Λ,
2. *the order of* $g_\perp$ *is* $M_\perp$,
3. *for any* $\mathsf{hk} \in \mathcal{K}$ *and* $x \in \bar{\mathcal{X}}$,

$$\mathsf{projkg}(\mathsf{hk} + \mathsf{hk}_\perp(x), x) = \mathsf{projkg}(\mathsf{hk}, x), \tag{4}$$
$$\mathsf{hash}(\mathsf{hk}_\perp(x), x) = g_\perp \ . \tag{5}$$

Definition 11 (Translation Indistinguishability). *A key-homomorphic SPHF* $\mathsf{PHF} = (\mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$ *is* $(\mathsf{hk}_\perp, M_z, \epsilon_{\mathrm{ti}})$*-translation indistinguishable for*

- *a function* $\mathsf{hk}_\perp : \bar{\mathcal{X}} \to \mathbb{G}$,
- *a positive natural number* $M_z \in \mathbb{N}_{>0}$,
- *a real* $0 \leq \epsilon_{\mathrm{ti}} \leq 1$,

if for any $z \in \{-M_z, \ldots, M_z\}$ *and for any* $\bar{x} \in \bar{\mathcal{X}}$,

$$\Delta(\mathsf{hashkg}(\Lambda), \mathsf{hashkg}(\Lambda) + z \cdot \mathsf{hk}_\perp(\bar{x})) \leq \epsilon_{\mathrm{ti}} \ , \tag{6}$$

where $\Delta(\cdot,\cdot)$ *is the statistical distance.*

Definition 12 (CPA-friendliness). *A SPHF* $\mathsf{PHF} = (\mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$ *is*

$$(\mathsf{hk}_\perp, g_\perp, M_\perp, M_z, \epsilon_{\mathrm{ti}})\text{-}\mathit{CPA}\text{-}\mathit{friendly}$$

for

- *a function* $\mathsf{hk}_\perp : \bar{\mathcal{X}} \to \mathbb{G}$,
- *two positive natural number* $M_z, M_\perp \in \mathbb{N}_{>0}$,
- *a group element* $g_\perp \in \mathbb{G}$,
- *a real* $0 \leq \epsilon_{\mathrm{ti}} \leq 1$,

if it is key-homomorphic, $(\mathsf{hk}_\perp, g_\perp, M_\perp)$*-strongly diverse, and* $(\mathsf{hk}_\perp, M_z, \epsilon_{\mathrm{ti}})$*-translation indistinguishable.*

Tag-based SPHF and CCA-friendliness. A tag-based SPHF [4] is a variant of SPHF from Definition 5 that include a tag from a set Tag during hashing and projection hashing. The syntax is given below. Let $\mathbf{P}$ be a subset membership problem that is efficiently sampleable.

Definition 13 ([4]). *A SPHF* $\mathsf{PHF} = (\mathsf{hashkg}^{tb}, \mathsf{projkg}^{tb}, \mathsf{hash}^{tb}, \mathsf{projhash}^{tb})$ *is recalled below*

$\mathsf{hashkg}^{\mathsf{tb}}(\Lambda)$**:** *Given as input aninstance* $\Lambda \xleftarrow{\$} I_\lambda$, *where* $\Lambda = (\mathcal{U}_\lambda, \mathcal{X}_\lambda, D(\mathcal{X}_\lambda), D(\mathcal{U}_\lambda \backslash \mathcal{X}_\lambda), \mathcal{Y}_\lambda, \mathsf{Rel}_\lambda)$, *and output a key* $\mathsf{hk} \in \mathcal{K}$.

$\mathsf{projkg}^{\mathsf{tb}}(\mathsf{hk}, x)$**:** *Given a hash key* $\mathsf{hk} \in \mathcal{K}$ $x \in \mathcal{X}_\lambda$ *with respect to* Λ, *output a projection key* pjk.

$\mathsf{hash}^{\mathsf{tb}}(\mathsf{hk}, x, \mathsf{tag})$**:** *Given as inputs a hash key* $\mathsf{hk} \in \mathcal{K}$, *a tag* $\mathsf{tag} \in \mathsf{Tag}$, *and* $x \in \mathcal{U}_\lambda$ *with respect to* Λ, *output a hash value* hval.

$\mathsf{projhash}^{\mathsf{tb}}(\mathsf{pjk}, x, y, \mathsf{tag})$**:** *Given a projection key* pjk, *a tag* $\mathsf{tag} \in \mathsf{Tag}$, *and* $x \in \mathcal{X}_\lambda$, $y \in \{0,1\}^{poly(\lambda)}$, *output a projected hash value* pjhval.

We require a form of tag-based completeness as follows. A SPHF $\mathsf{tPHF} = (\mathsf{hashkg}^{\mathsf{tb}}, \mathsf{projkg}^{\mathsf{tb}}, \mathsf{hash}^{\mathsf{tb}}, \mathsf{projhash}^{\mathsf{tb}})$ is *complete* if for all $\lambda \in \mathbb{N}$, all $\Lambda \xleftarrow{\$} I_\lambda$ specifying $\Lambda = (\mathcal{U}_\lambda, \mathcal{X}_\lambda, D(\mathcal{X}_\lambda), D(\mathcal{U}_\lambda \setminus \mathcal{X}_\lambda), \mathcal{Y}_\lambda, \mathsf{Rel}_\lambda)$, all $\mathsf{hk} \leftarrow \mathsf{hashkg}^{\mathsf{tb}}(\Lambda)$, for all $\mathsf{tag} \in \mathsf{Tag}$, all $x \in \mathcal{X}_\lambda$ and $y \in \mathcal{Y}_\lambda$ such that $(x, y) \in \mathsf{Rel}_\lambda$, if $\mathsf{pjk} \leftarrow \mathsf{projkg}^{\mathsf{tb}}(\mathsf{hk}, x)$, then $\mathsf{hash}^{\mathsf{tb}}(\mathsf{hk}, x, \mathsf{tag}) = \mathsf{projhash}^{\mathsf{tb}}(\mathsf{pjk}, x, y, \mathsf{tag})$.

A form of projection-key homomorphism is required for the tag-based SPHF, towards CCA-friendliness. We note that the property is adapted for the SPHF variant from [43], where the the projection key generation takes an instance as input as well. A similar property in [18, Def. 7] treats earlier a stronger form of projection key homomorphism, meanwhile ours is weakened to receive the instance as input (see the discussion in [43, Sect. 3.2]).

Definition 14 (Projection Key Homomorphism). *A SPHF*

$$\mathit{PHF} = (\mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$$

over a subset membership problem $\mathbf{P}$ *is said to be* projection key-homomorphic *if the following holds:*

1. *The set* $\mathcal{K}$ *of hashing keys and the set* $\mathcal{K}^{\mathsf{pj}}$ *of projection keys are Abelian groups.*
2. *Fixing any instance* $\Lambda \xleftarrow{\$} I_\lambda$ *and a statement* $x \in \mathcal{U}$ *in the universe of* Λ*, the projection* $\mathsf{hashkg}(\cdot, x)$ *of PHF is a group homomorphism from* $\mathcal{K}$ *to* $\mathcal{K}^{\mathsf{pj}}$*.*

The remain properties to complete the definition of CCA-friendliness are summarized in the following definitions.

Definition 15 (2-universality). *A key-homomorphic tag-based SPHF* $\mathit{tPHF} = (\mathsf{hashkg}^{tb}, \mathsf{projkg}^{tb}, \mathsf{hash}^{tb}, \mathsf{projhash}^{tb})$ *for a subset membership problem* $\mathbf{P}$ *is* $\epsilon^{\mathsf{tb}}_{2\mathrm{u}}$-2-universal *if for any* $\Lambda \xleftarrow{\$} I_\lambda$ *and any* $x \in \mathcal{U}$*, and any* $\hat{x} \in \mathcal{U} \setminus \mathcal{X}$*, for any* $\mathsf{tag} \neq \mathsf{tag}' \in \mathsf{Tag}$*, for any* $\mathsf{pjk}^{tb} \in \mathcal{K}^{\mathsf{pj}}$ *and any hash values* $\mathsf{hval}^{tb}, \mathsf{hval}^{tb'}$

$$\Pr_{\mathsf{hk}^{tb}}\left[\begin{array}{l}\mathsf{hval}^{tb} \leftarrow \mathsf{hash}^{tb}(\mathsf{hk}, x, \mathsf{tag})\\ \mathsf{hval}^{tb'} \leftarrow \mathsf{hash}^{tb}(\mathsf{hk}, \hat{x}, \mathsf{tag}')\\ \mathsf{pjk}^{tb} \leftarrow \mathsf{projkg}^{tb}(\mathsf{hk}, x)\end{array}\right] \leq \epsilon^{tb}_{2\mathrm{u}} \cdot \Pr_{\mathsf{hk}^{tb}}\left[\begin{array}{l}\mathsf{hval}^{tb} \leftarrow \mathsf{hash}^{tb}(\mathsf{hk}, x, \mathsf{tag})\\ \mathsf{pjk}^{tb} \leftarrow \mathsf{projkg}^{tb}(\mathsf{hk}, x)\end{array}\right]$$

and the probability is taken over the choice of $\mathsf{hk}^{tb} \xleftarrow{\$} \mathsf{hashkg}^{tb}(\Lambda)$*.*

Definition 16 (Universal Translation Indistinguishability). *A key-homomorphic tag-based SPHF* $\mathit{tPHF} = (\mathsf{hashkg}^{tb}, \mathsf{projkg}^{tb}, \mathsf{hash}^{tb}, \mathsf{projhash}^{tb})$ *is* $(\widetilde{\mathsf{hashkg}}^{tb}, M_z, \epsilon^{tb}_{\mathrm{ti}})$*-universally-translation indistinguishable for an algorithm* $\widetilde{\mathsf{hashkg}}^{tb}$ *that receives* Λ *and outputs* hk^{tb} *in a subset of the hash key space* $\mathcal{K}$*, and for* $M_z \in \mathbb{N}_{>0}$*, if*

$$\forall z \in [-M_z, M_z] : \Delta(\mathsf{hashkg}^{tb}(\Lambda), \mathsf{hashkg}^{tb}(\Lambda) + z \cdot \widetilde{\mathsf{hashkg}}^{tb}(\Lambda)) \leq \epsilon^{tb}_{\mathrm{ti}} \ .$$

Definition 17 (CCA-friendliness). *A tag-based SPHF*

$$\mathit{tPHF} = (\mathsf{hashkg}^{tb}, \mathsf{projkg}^{tb}, \mathsf{hash}^{tb}, \mathsf{projhash}^{tb})$$

is $(\widetilde{\mathsf{hashkg}}^{tb}, \mathit{Coeff}, \epsilon^{tb}_{2\mathrm{u}}, M_z, \epsilon^{tb}_{\mathrm{ti}})$*-CCA-friendly for*

- *an algorithm* $\widetilde{\mathsf{hashkg}}^{tb}$ *that receives* Λ *and outputs* hk^{tb} *in a subset of the hash key space* $\mathcal{K}$*,*
- *a positive natural number* $M_z \in \mathbb{N}_{>0}$*,*
- *a subset* $\mathit{Coeff} \subseteq \mathbb{Z}$ *of coefficients,*

if it is projection key-homomorphic, $\epsilon^{tb}_{2\mathrm{u}}$*-2-universal,* $(\widetilde{\mathsf{hashkg}}^{tb}, M_z, \epsilon^{tb}_{\mathrm{ti}})$*-universall-translation indistinguishable, and for all* $c \in \mathit{Coeff}$*, the tag-based SPHF* $(c \cdot \widetilde{\mathsf{hashkg}}^{tb}, \mathsf{projkg}^{tb}, \mathsf{hash}^{tb}, \mathsf{projhash}^{tb})$ *where* $c \cdot \widetilde{\mathsf{hashkg}}^{tb}$ *denotes the algorithm that runs* $\widetilde{\mathsf{hashkg}}^{tb}$ *then multiplies the output hash key by* c*.*

Putting fowards our building block, a tag-based $\mathsf{tPHF} = (\mathsf{hashkg}^{\mathsf{tb}}, \mathsf{projkg}^{\mathsf{tb}}, \mathsf{hash}^{\mathsf{tb}}, \mathsf{projhash}^{\mathsf{tb}})$, that meets the requirements to provide CCA-security for our MIFE in Figure 4 of Section 5.1, can be instantiated by [35, 4] under DDH, or [18, Lemma 3] under DCR. The proofs for FE-CCA-friendliness for following Definition 17 can be found in [18, Lemma 27, Lemma 29], with syntactical modifications taking into account that that our projection hash keygen receives a statement $x \in \mathcal{U}$ as input[14]. We refer to our later Section 5.2 for more details.

[14] The FE-CCA-friendly tag-based tPHF from [4] can be put into our tag-based SPHF syntax where its $\mathsf{projkg}^{\mathsf{tb}}$ ignores the statement. It is more the Gennaro-Lindell SPHF [43] whose projkg is of crucial importance to rely on the statement, and we do *not* require FE-CCA-friendliness from Gennaro-Lindell SPHF.

2.3 Σ-Protocol

Let R be an NP relation with statements x and witnesses w. We denote by $\mathscr{L}_{\mathsf{R}} = \{x \mid \exists w \text{ s.t. } (x,w) \in \mathsf{R}\}$ the language induced by R. A Σ-protocol for an NP relation R for language $\mathscr{L}_{\mathsf{R}}$ is a tuple of PPT algorithms $\mathbf{\Sigma} = (\mathsf{Init}, \mathsf{Chall}, \mathsf{Resp}, \mathsf{Verify})$ such that

- $\mathsf{Init}(x,w)$: given a statement $x \in \mathscr{L}_{\mathsf{R}}$, and a witness w such that $(x,w) \in \mathsf{R}$, outputs a first flow message (*i.e.*, commitment) Ω and a state st, where we assume st includes x, w,
- $\mathsf{Chall}()$: samples a challenge $\gamma \xleftarrow{\$} \mathcal{CH}$ (without taking any input),
- $\mathsf{Resp}(\mathsf{st}, \gamma)$: given a state st and a challenge $\gamma \in \mathcal{CH}$, outputs a third flow message (*i.e.*, response) τ,
- $\mathsf{Verify}(x, \Omega, \gamma, \tau)$: given a statement $x \in \mathscr{L}_{\mathsf{R}}$, a commitment Ω, a challenge $\gamma \in \mathcal{CH}$, and a response τ, outputs a bit $b \in \{0,1\}$.

Definition 18 (Correctness). *A Σ-protocol is* correct*, if for all $(x,w) \in \mathsf{R}$, $(\Omega, \mathsf{st}) \leftarrow \mathsf{Init}(x,w)$, $\gamma \in \mathcal{CH}$, and $\tau \leftarrow \mathsf{Resp}(\mathsf{st},\gamma)$, it holds that $\mathsf{Verify}(x,\Omega,\gamma,\tau) = 1$.*

Definition 19 (High Min-Entropy). *A Σ-protocol has* high min-entropy *if for all $(x,w) \in \mathsf{R}$ and (possibly unbounded) adversary $\mathcal{A}$, it holds that*

$$\Pr[(\Omega, \mathsf{st}) \leftarrow \mathsf{Init}(x,w), \Omega' \leftarrow \mathcal{A}(1^n) : \Omega = \Omega'] = \mathrm{negl}(n).$$

Definition 20 (Non-abort HVZK). *A Σ-protocol is* non-abort honest-verifier zero-knowledge *(HVZK), if there exists a PPT zero-knowledge simulator Sim such that the distributions of $\mathsf{Sim}(x,\gamma)$ and the honestly generated transcript with Init initialized with (x,w) are statistically indistinguishable for any $x \in \mathscr{L}_{\mathsf{R}}$, and $\gamma \in \mathcal{CH}$, where the honest execution is conditioned on γ being used as the challenge and no abort occurring.*

We write HVZK for short if the Σ-protocol never aborts.

Definition 21 (k-Special Soundness). *A Σ-protocol is k-*special sound*, if there exists a* deterministic *PT extractor Ext such that given k valid transcripts $\{(\Omega, \gamma_i, \tau_i)\}_{i \in [k]}$ for statement x with pairwise distinct challenges $(\gamma_i)_i$, outputs a witness w such that $(x,w) \in \mathsf{R}$.*

2.4 Non-interactive zero knowledge proof systems

We will use *non-interactive zero-knowledge proofs* (NIZK) to prove the validity of the ciphertexts. Let $\mathcal{URS} = \{0,1\}^\ell$ be a set of *uniform random strings* for some $\ell \in \mathbb{N}$ and $\mathcal{SRS}$ be some set of *structured random strings* with efficient membership test. An NIZK for a relation Rel with common reference string space $\mathcal{CRS} = \mathcal{SRS} \times \mathcal{URS}$ is a tuple of PPT algorithms $(\mathsf{GenCRS}, \mathsf{Prove}^{\mathsf{H}}, \mathsf{Verify}^{\mathsf{H}})$, where the latter two are oracle-calling, such that:

- $\mathsf{GenCRS}(1^\lambda)$: outputs a structured reference string $\mathsf{srs} \in \mathcal{SRS}$,
- $\mathsf{Prove}^{\mathsf{H}}(\mathsf{crs}, x, w)$: receives a $\mathsf{crs} = (\mathsf{srs}, \mathsf{urs}) \in \mathcal{CRS}$, a statement x and a witness w, and outputs a proof π,
- $\mathsf{Verify}^{\mathsf{H}}(\mathsf{crs}, x, \pi)$: receives a $\mathsf{crs} = (\mathsf{srs}, \mathsf{urs}) \in \mathcal{CRS}$, a statement x and a proof π, and outputs a bit $b \in \{0,1\}$.

We recall that $\mathscr{L}_{\mathsf{Rel}} = \{x \mid \exists w \ : \ (x,w) \in \mathsf{Rel}\}$ denotes the language induced by Rel. If there is no crs needed, *i.e.* $\mathcal{CRS} = \varnothing$, we then omit crs as an input to Prove and Verify.

Definition 22 (Correctness). *An NIZK is* correct *if for any $\mathsf{crs} = (\mathsf{srs}, \mathsf{urs})$ with $\mathsf{srs} \leftarrow \mathsf{GenCRS}(1^\lambda)$ and $\mathsf{urs} \leftarrow \mathcal{URS}$, $(x,w) \in \mathsf{Rel}$, and $\pi \leftarrow \mathsf{Prove}^{\mathsf{H}}(\mathsf{crs}, x, w)$, it holds that $\mathsf{Verify}^{\mathsf{H}}(\mathsf{crs}, x, \pi) = 1$.*

Definition 23 (Zero-Knowledge). *An* NIZK *is* zero-knowledge *(ZK) if there exists a PPT simulator* $\mathsf{Sim} = (\mathsf{Sim}_{\mathsf{crs}}, \mathsf{Sim}_{\mathsf{H}}, \mathsf{Sim}_{\pi})$ *such that for any PPT adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathrm{zk}}_{\mathcal{A}}(\lambda) = \left| \Pr\begin{bmatrix} \mathsf{srs} \leftarrow \mathsf{GenCRS}(1^{\lambda}), \\ \mathsf{crs} = (\mathsf{srs}, \mathsf{urs}), \\ \mathcal{A}^{\mathsf{H},\mathcal{P}}(\mathsf{crs}) = 1 \end{bmatrix} - \Pr\begin{bmatrix} \mathsf{crs} \leftarrow \mathsf{Sim}_{\mathsf{crs}}(1^{\lambda}), \\ \mathsf{crs} = (\mathsf{srs}, \mathsf{urs}), \\ \mathcal{A}^{\mathsf{Sim}_{\mathsf{H}},\mathcal{S}}(\mathsf{crs}) = 1 \end{bmatrix} \right| = \mathrm{negl}(\lambda),$$

where $\mathcal{P}$ *and* $\mathcal{S}$ *are oracles that on input* (x, w) *return* $\perp$ *if* $(x, w) \notin \mathsf{Rel}$*, and else output* $\mathsf{Prove}^{\mathsf{H}}(\mathsf{crs}, x, w)$ *or* $\mathsf{Sim}_{\pi}(\mathsf{crs}, x)$ *respectively. Note that the probability is taken over the randomness of* Sim *and* $\mathcal{A}$*, and the random choices of* H *and* urs*. Also,* $\mathsf{Sim}_{\mathsf{crs}}, \mathsf{Sim}_{\mathsf{H}}$ *and* Sim_{π} *have a shared state.*

We define different notions of soundness. We remark that the soundness relation $\widetilde{\mathsf{Rel}}$ can be different from the (correctness) relation Rel. If $\widetilde{\mathsf{Rel}}$ is not explicitly defined, we implicitly set $\widetilde{\mathsf{Rel}} = \mathsf{Rel}$.

Finally, we also recall the definition of *simulation soundness* below.

Definition 24. *An* NIZK *is* simulation-sound *for relation* $\mathsf{EqMsgRel}^{\mathsf{sk}}$ *if there exists PPT simulators* SimCRS *and* $(\mathsf{Sim}_{\mathsf{H}}, \mathsf{Sim}_{\pi})$ *so that for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*, it holds that*

CRS Indistinguishability. *For any PPT adversary* $\mathcal{A}$*, the following advantage*

$$\mathsf{Adv}^{\mathrm{crs}}_{\mathcal{A}}(\lambda) = \left| \Pr\begin{bmatrix} \mathsf{srs} \leftarrow \mathsf{GenCRS}(1^{\lambda}), \mathsf{urs} \leftarrow \mathcal{URS}, \\ \mathsf{crs} = (\mathsf{srs}, \mathsf{urs}) : \mathcal{A}^{\mathsf{H}}(\mathsf{crs}) = 1 \end{bmatrix} - \Pr\begin{bmatrix} (\overline{\mathsf{crs}}, \mathsf{td}) \leftarrow \mathsf{SimCRS}(1^{\lambda}) \\ s.t.\ \mathcal{A}^{\mathsf{H}}(\overline{\mathsf{crs}}) = 1 \end{bmatrix} \right|$$

is negligible in λ.

Simulation-Soundness. *For any PPT adversary* $\mathcal{A}$*, the following advantage*

$$\mathsf{Adv}^{\mathrm{s\text{-}snd}}_{\mathcal{A}}(\lambda) = \Pr\begin{bmatrix} \mathsf{srs} \leftarrow \mathsf{SimCRS}(1^{\lambda}), \mathsf{urs} \leftarrow \mathcal{URS}, & \\ \overline{\mathsf{crs}} = (\mathsf{srs}, \mathsf{urs}) : \mathcal{A}^{\mathsf{Sim}_{\mathsf{H}}}(\overline{\mathsf{crs}}) \rightarrow (x, \mathsf{st}) & : x' \notin \mathscr{L}_{\mathsf{EqMsgRel}} \wedge \pi' \notin \mathcal{Q}_{\mathsf{Sim}_{\pi}} \\ \mathcal{A}_1^{\mathsf{Sim}_{\pi}(\overline{\mathsf{crs}}, \mathsf{td}, \cdot, \mathsf{st})}(\overline{\mathsf{crs}}, \mathsf{st}) = (x', \pi') & \wedge\ \mathsf{Verify}^{\mathsf{Sim}_{\mathsf{H}}}(\overline{\mathsf{crs}}, x', \pi') = 1 \\ \textit{with } \mathsf{Sim}_{\pi}(\overline{\mathsf{crs}}, \mathsf{td}, x, \mathsf{st}) = \pi & \end{bmatrix},$$

where $\mathcal{Q}_{\mathsf{Sim}_{\pi}}$ *contains proofs that resulted from simulation queries to* Sim_{π} *by* $\mathcal{A}$*, is negligible in* λ*.*

Fiat-Shamir transformation. We recall the Fiat-Shamir transformation [41] to turn a Σ-protocol into a NIZK. The transformation gives a NIZK that is correct and satisfies adaptive knowledge soundness. Moreover, applying the result of [40] in the case the Σ-protocol is 2-special sound, we obtain a NIZK that is also simulation-sound. Sometimes, we require more involved variants of this transformations. In that case, we provide the compiled NIZK explicitly.

Theorem 25. *Let* $\mathsf{\Sigma} = (\mathsf{Init}, \mathsf{Chall}, \mathsf{Resp}, \mathsf{Verify})$ *be a* Σ*-protocol that satisfies* correctness*,* high-min entropy*,* honest verifier zero-knowledge*, and* 2-Special Soundness*. The* Fiat-Shamir transformation $FS[\mathsf{\Sigma}] = (\mathsf{GenCRS}, \mathsf{Prove}^{\mathsf{H}}, \mathsf{Verify}^{\mathsf{H}})$ *is described below:*

- $\mathsf{GenCRS}(1^n)$*: outputs the empty string* ϵ *as we do not require a common reference string and omit* crs *as an input for other below algorithms,*
- $\mathsf{Prove}^{\mathsf{H}}(x, w)$*: receives a statement* x *and a witness* w*, runs* $(\Omega, \mathsf{st}) \leftarrow \mathsf{Init}(x, w)$*, computes the challenge* $\gamma \leftarrow \mathsf{H}(x, \Omega)$*, then computes* $\tau \leftarrow \mathsf{Resp}(\mathsf{st}, \gamma)$ *and outputs* $\pi = (\Omega, \gamma, \tau)$*.*
- $\mathsf{Verify}^{\mathsf{H}}(x, \pi)$*: receives a statement* x *and a proof* $\pi = (\Omega, \gamma, \tau)$*, and outputs* $b \leftarrow \mathsf{Verify}(x, \Omega, \gamma, \tau) \wedge \gamma = \mathsf{H}(x, \Omega)$*.*

In the ROM, $FS[\mathsf{\Sigma}]$ *is a* NIZK *that is* correct *and satisfies* adaptive knowledge soundness*. If* $\mathsf{\Sigma}$ *is 2-special sound, then* $FS[\mathsf{\Sigma}]$ *is also* simulation-sound*.*

2.5 Function Classes for Functional Encryption

Similar to the work of [67], we define classes of *functions with public inputs* that will be considered for our MCFE generic construction. As already studied in [67], the main interest of such public inputs gives rise to access control over functional keys.

Definition 26 (Functions with public inputs). *Let $\lambda, n \in \mathbb{N}$ and let $\mathcal{D}_{\lambda,i}$ and $\mathcal{R}_\lambda$ be domains and ranges indexed by λ in some ensembles $\{\mathcal{D}_{\lambda,i}\}_\lambda$ where $i \in [n]$, $\{\mathcal{R}_\lambda\}_\lambda$, respectively. A* function class $\mathcal{F} = \{F_{\lambda,n}\}_{\lambda,n}$ with public inputs $(\mathcal{Z}_{\lambda,i})_{i\in[n]}$*, where $\mathcal{Z}_{\lambda,i} := \{0,1\}^{poly(\lambda)}$, is defined to contain $F_{\lambda,n} : \prod_{i=1}^{n} (\mathcal{D}_{\lambda,i} \times \mathcal{Z}_{\lambda,i}) \to \mathcal{R}_\lambda$.*

In the following the index n is a function in λ and we omit it for clarity. In particular, by including public inputs, we can treat the integration of access control. Finally, we recall the function class to compute inner products that will be considered for our MIFE construction.

Definition 27 (Inner Products). *We consider the functionality $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$ that contains $F_{\mathbf{y}_1,\dots,\mathbf{y}_n} : \prod_{i\in[n]} \left(\mathbb{Z}_q^{N_i}\right) \to \mathbb{Z}_q$ defined as $F_{\mathbf{y}_1,\dots,\mathbf{y}_n}(\mathbf{x}_1,\dots,\mathbf{x}_n) := \sum_{i=1}^{n} \langle \mathbf{x}_i, \mathbf{y}_i \rangle$, whose inputs $\mathbf{x}_i$ and parameters $\mathbf{y}_i$ satisfy $\max(\|\mathbf{x}_i\|_\infty, \|\mathbf{y}_i\|_\infty) < B$, with $B = \mathsf{poly}(\lambda) \in \mathbb{N}$ being a polynomial.*

3 CCA-Security for Multi-Client FE: Definitions

We procceed by giving the *IND-CCA* security game in Figure 3, followed by the adapted notion of *admissible adversaries*. In the following definitions we use the specific function class that allow public inputs for the ease of presentation when instantiating concretely later. Of course all notions can be generalized to general function classes. The syntax of MCFE with public inputs is recalled in Definition 28.

Definition 28 (Multi-client functional encryption with public input). *A* multi-client functional encryption (MCFE) scheme with public inputs*, for the class $\mathcal{F}$ with public inputs $(\mathcal{Z}_{\lambda,i})_{i\in[n]}$ where $\mathcal{Z}_{\lambda,i} := \mathsf{Tag} \times \widetilde{\mathcal{Z}}_{\lambda,i}$ for some set $\mathsf{Tag} = \{0,1\}^{poly(\lambda)}$, consists of four algorithms* $(\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$*:*

$\mathsf{Setup}(1^\lambda, 1^n)$: *Given as inputs 1^λ for a security parameter λ, and a number of clients n, output some public paramters pp, a master secret key msk and n encryption keys $(\mathsf{ek}_i)_{i\in[n]}$.*

$\mathsf{Extract}(\mathsf{msk}, F_\lambda)$: *Given a function description $F_\lambda : \prod_{i=1}^{n} (\mathcal{D}_{\lambda,i} \times \mathcal{Z}_{\lambda,i}) \to \mathcal{R}_\lambda$ in $\mathcal{F}$, and the master secret key msk, output a decryption key dk_{F_λ}.*

$\mathsf{Enc}(\mathsf{ek}_i, x_i, z_i)$: *Given as inputs* public *data $z_i = (\mathsf{tag}, \tilde{z}_i) \in \mathcal{Z}_{\lambda,i}$ that contains some tag, an encryption key ek_i, a message $x_i \in \mathcal{D}_{\lambda,i}$, output a ciphertext $(\mathsf{ct}_{\mathsf{tag},i}, z_i)$. For a specific client i, the sets $\mathcal{D}_{\lambda,i}$ and $\mathcal{Z}_{\lambda,i}$ are indexed by λ in some ensembles $\{\mathcal{D}_{\lambda,i}\}_\lambda, \{\mathcal{Z}_{\lambda,i}\}_\lambda$.*

$\mathsf{Dec}(\mathsf{dk}_{F_\lambda}, \mathbf{c})$: *Given the decryption key dk_{F_λ} and a vector of ciphertexts $\mathbf{c} := (\mathsf{ct}_{\mathsf{tag},i}, z_i)_i$ of length n, output an element in $\mathcal{R}_\lambda$ or an invalid symbol $\perp$.*

Correctness. For sufficiently large $\lambda \in \mathbb{N}$, for all $(\mathsf{msk}, (\mathsf{ek}_i)_{i\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda)$, all functions $F_{\lambda,n} : \prod_i (\mathcal{D}_{\lambda,i} \times \mathcal{Z}_{\lambda,i}) \to \mathcal{R}_\lambda$ and $\mathsf{dk}_{F_{\lambda,n}} \leftarrow \mathsf{Extract}(\mathsf{msk}, F_{\lambda,n})$, for all $\mathsf{tag} \in \mathsf{Tag}$ and $(z_i)_{i=1}^n \in \mathcal{Z}_{\lambda,1} \times \cdots \times \mathcal{Z}_{\lambda,n}$, for all $(x_i)_{i\in[n]} \in \mathcal{D}_{\lambda,1} \times \cdots \times \mathcal{D}_{\lambda,n}$, if $F_\lambda((x_i, z_i)_i) \neq \perp$ and $z_i = (\mathsf{tag}, \tilde{z}_i) \in \mathcal{Z}_i$ for all i, the following holds with overwhelming probability:

$$\mathsf{Dec}\left(\mathsf{dk}_{F_\lambda}, (\mathsf{Enc}(\mathsf{ek}_i, x_i, z_i))_{i\in[n]}\right) = F_{\lambda,n}((x_i, z_i)_i)$$

where the probability is taken over the random coins of the algorithms.

Definition 29 (CCA-admissible adversaries with public inputs). *Let $\mathcal{A}$ be a* ppt *adversary and let $\mathcal{E} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$ be* an MCFE scheme with public inputs *for the function class $\mathcal{F}$ with public inputs $\mathcal{Z}_{\lambda,i} := \mathsf{Tag} \times \widetilde{\mathcal{Z}}_{\lambda,i}$. In the security game given in Figure 3 for $\mathcal{A}$ considering $\mathcal{E}$, let the sets $(\mathcal{C}, \mathcal{Q}_{\mathsf{Dec}}, \mathcal{Q}_{\mathsf{dk}}, \mathcal{H})$ be the sets of corrupted clients,* decryption *queries, functional key queries, and honest clients, in that order. We say that $\mathcal{A}$* is NOT admissible w.r.t *$(\mathcal{C}, \mathcal{Q}_{\mathsf{Dec}}, \mathcal{Q}_{\mathsf{dk}}, \mathcal{H})$ if any of the following conditions holds:*

```
Initialise$(1^\lambda, 1^n)$ $\boxed{\textbf{Initialise}(1^\lambda, (x_i^{(0)}, x_i^{(1)})_{i\in[n]})}$
    $b \xleftarrow{\$} \{0,1\}$
    $(\mathsf{pp}, \mathsf{msk}, (\mathsf{ek}_i)_{i\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$
    $\mathcal{Q}_{\mathsf{dk}}, \mathcal{Q}_{\mathsf{Dec}} := \varnothing,\ \mathcal{C} := \varnothing,\ \mathcal{H} := [n]$
    Return $\mathsf{pp}$

Enc$(i, x_i, (\mathsf{tag}, \tilde{z}_i))$
    Return $\mathsf{Enc}(\mathsf{ek}_i, x_i, (\mathsf{tag}, \tilde{z}_i))$

Dec$(F, \mathbf{c})$
    Run $\mathsf{dk}_F \leftarrow \mathsf{Extract}(\mathsf{msk}, F)$
    $\mathcal{Q}_{\mathsf{Dec}} := \mathcal{Q}_{\mathsf{Dec}} \cup \{(F, \mathbf{c})\}$
    Return $\mathsf{Dec}(\mathsf{dk}_F, \mathbf{c})$

LoR$(i, x_i^{(0)}, x_i^{(1)}, (\mathsf{tag}^*, \tilde{z}_i^{(chal)}))$ $\boxed{\textbf{LoR}(i, (\mathsf{tag}^*, \tilde{z}_i^{(chal)}))}$
    $\mathsf{Enc}(\mathsf{ek}_i, x_i^{(b)}, (\mathsf{tag}^*, \tilde{z}_i^{(chal)})) \to \mathsf{ct}^{(b)}_{\mathsf{tag}^*,i}$
    Return $\mathsf{ct}^{(b)}_{\mathsf{tag}^*,i}$

Corrupt$(i)$
    $\mathcal{C} := \mathcal{C} \cup \{i\}$
    $\mathcal{H} := \mathcal{H} \setminus \{i\}$
    Return $\mathsf{ek}_i$

Extract$(F)$
    $\mathcal{Q}_{\mathsf{dk}} := \mathcal{Q}_{\mathsf{dk}} \cup \{F\}$
    $\mathsf{dk}_F \leftarrow \mathsf{Extract}(\mathsf{msk}, F)$
    Return $\mathsf{dk}_F$

Finalise$(b')$
    If $\mathcal{A}$ is NOT admissible w.r.t $(\mathcal{C}, \mathcal{Q}_{\mathsf{Dec}}, \mathcal{Q}_{\mathsf{dk}}, \mathcal{H})$:
        return 0
    Else return $\left(b' \stackrel{?}{=} b\right)$
```

Fig. 3: The security games $\mathsf{Expr}^{\mathsf{mc\text{-}ind\text{-}cca}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$, $\boxed{\mathsf{Expr}^{\mathsf{mc\text{-}sel\text{-}ind\text{-}cca}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)}$ for Definition 30

1. *There exist* $(F, \mathbf{c}) \in \mathcal{Q}_{\mathsf{Dec}}$ *such that there exists* $i \in \mathcal{H}$ *and* $\mathbf{c}[i] = \mathsf{ct}^{(b)}_{\mathsf{tag}^*,i}$.
2. *There exist* $\mathsf{tag} \in \mathsf{Tag}$, *a function* $F \in \mathcal{F}$, *two challenges* $(x_i^{(0)}, x_i^{(1)})_{i\in[n]}$, *public inputs* $\tilde{z}_i^{(chal)} \in \widetilde{\mathcal{Z}}_{\lambda,i}$, *such that* $F \in \mathcal{Q}$ *is queried to* **Extract**, *and there exist vectors* $(\mathbf{t}^{(0)}, \mathbf{t}^{(1)}, \mathbf{v}^{(chal)})$ *so that* $\forall\, i \in \mathcal{H} : \mathbf{t}^{(b)}[i] = x_i^{(b)}$ *and* $\mathbf{v}^{(chal)}[i] = \tilde{z}_i^{(chal)}$ *satisfying*

$$F(\mathbf{t}^{(0)}, (\mathsf{tag}, \mathbf{v}[i])_{i\in[n]}) \neq F(\mathbf{t}^{(1)}, (\mathsf{tag}, \mathbf{v}[i])_{i\in[n]}) \ , \tag{7}$$

Otherwise, we say that $\mathcal{A}$ is admissible w.r.t $(\mathcal{C}, \mathcal{Q}_{\mathsf{Dec}}, \mathcal{Q}_{\mathsf{dk}}, \mathcal{H})$.

Definition 30 (CCA-security with repetitions for MCFE with public inputs). *An MCFE scheme* $\mathcal{E} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$ *for the function class* $\mathcal{F}$ *with public inputs is* CCA-secure *if for all* ppt *adversaries* $\mathcal{A}$, *and for all sufficiently large* $\lambda \in \mathbb{N}$, *the following probability is negligible*

$$\mathsf{Adv}^{\mathsf{mc\text{-}w\text{-}rep}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) := \left| \Pr[\mathsf{Expr}^{\mathsf{mc\text{-}ind\text{-}cca}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) = 1] - \frac{1}{2} \right| \ .$$

The game $\mathsf{Expr}^{\mathsf{mc\text{-}ind\text{-}cca}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ *is depicted in Figure 3. The probability is taken over the random coins of* $\mathcal{A}$ *and the algorithms.*

Weaker notions can be found in Appendix A.5.

Remark 31. (Initial thoughts between CCA and CPA) In the definition of admissible adversaries with respect to CCA security, we do *not* enforce that the adversary query to the decryption oracle only on functions that they do *not* have the functional key for. At first glance, this does allow a meaningful capturing of the CPA security notion from Definition 29: any CPA adversary, when playing the CCA game, can call the decryption oracle **Dec** on any ciphertext of their choice instead of doing the decryption themselves, using the function that they ask for the functional key. The more interesting cases that distinguish CCA from CPA concern the functions for which there are not keys in the adversary's possession, and even taking into account the further attempts to malleate the ciphertexts before submitting to **Dec**. Finally, as it is already mentioned in [18] for the single-client setting, the **Dec** oracle is *stronger* than the key extraction oracle **Extract** itself, given that we do not allow decryption queries on the challenge components. Therefore, the **Dec** oracle does *not* need to put constraints on the input function, though the adversary have not the corresponding functional key.

Relation to the single-client setting. In the definition of *CCA-admissible adversaries* for MCFE with public inputs, we employ the strong notion of admissibility from [66], particularly the condition Equation (7). The main strength of this stronger notion, in comparison to existing works that follow the seminal [30], is that it allows deriving *public-key* single-client FE schemes from a *secret-key* MCFE, via the power of corruption. The following lemma shows that the same transformation can be done for the *CCA-security* notion. Intuitively, because CCA-secure in public-key setting is strictly stronger than CPA-security, our model for IND-CCA security is at least as strong as the state-of-the-art for IND-CPA security in the MCFE setting. The proof is given in Appendix F.

Lemma 32. *Let $\mathcal{F}$ be a function class with public inputs $(\mathcal{Z}_{\lambda,i})_{i\in[n]}$ where $\mathcal{Z}_{\lambda,i} \coloneqq \mathsf{Tag} \times \widetilde{\mathcal{Z}}_{\lambda,i}$ for some tag space $\mathsf{Tag} = \{0,1\}^{poly(\lambda)}$. The elements of $\mathcal{F}$ are $F_{\lambda,n} : \prod_{i=1}^{n} (\mathcal{D}_{\lambda,i} \times \mathcal{Z}_{\lambda,i}) \to \mathcal{R}_\lambda$. Suppose that $\mathcal{F}$ contains the identity function $F^{\mathsf{id}}_{\lambda,n}$ where for all $(x_i, z_i)_i$, $F^{\mathsf{id}}_{\lambda,n}((x_i, z_i)_i) = (x_i, z_i)_i$. If there exists a* CCA-secure *MCFE scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$ for the function class $\mathcal{F}$ with public inputs, then there exists a* CCA-secure *FE scheme for the same function class $\mathcal{F}$.*

4 CCA-Security for Multi-Client FE: Constructions

We turn our attention to building IND-CCA secure MCFE for function classes with auxiliary inputs *as per* Definition 30. The definitions of NIZK are recalled in Section 2. In the below contents we use some common notation : for the class $\mathcal{F}$ with public inputs $(\mathcal{Z}_{\lambda,i})_{i\in[n]}$ where $\mathcal{Z}_{\lambda,i} \coloneqq \mathsf{Tag} \times \widetilde{\mathcal{Z}}_{\lambda,i}$ for some tag space $\mathsf{Tag} = \{0,1\}^{poly(\lambda)}$, let

$$\mathsf{MCFE}^{\mathsf{cpa}}[\mathcal{F}, (\mathcal{Z}_{\lambda,i})_{i\in[n]}] = (\mathsf{Setup}^{\mathsf{cpa}}, \mathsf{Extract}^{\mathsf{cpa}}, \mathsf{Enc}^{\mathsf{cpa}}, \mathsf{Dec}^{\mathsf{cpa}})$$

be an IND-CPA MCFE.

4.1 Bootstrapping to CCA-secure for MCFE from Commitments and NIZKs

We start from $\mathsf{MCFE}^{\mathsf{cpa}}[\mathcal{F}, (\mathcal{Z}_{\lambda,i})_{i\in[n]}]$ that is an *IND-CPA* secure MCFE. We denote by $(\mathsf{Setup}^{\mathsf{cpa}}, \mathsf{Extract}^{\mathsf{cpa}}, \mathsf{Enc}^{\mathsf{cpa}}, \mathsf{Dec}^{\mathsf{cpa}})$ the algorithms of $\mathsf{MCFE}^{\mathsf{cpa}}[\mathcal{F}, (\mathcal{Z}_{\lambda,i})_{i\in[n]}]$. Let $\mathsf{EECom} = (\mathsf{Setup}, \mathsf{SetupTr}, \mathsf{Commit}, \mathsf{ExtCom}, \mathsf{OpenCom}, \mathsf{SimCom}, \mathsf{Verify})$ be an equivocable-extractable commitment scheme. Let $\mathsf{Com} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify})$ be a perfectly binding and computationally hiding commitment scheme. Let $\mathsf{NIZK} = (\mathsf{GenCRS}, \mathsf{Prove}^{\mathsf{H}}, \mathsf{Verify}^{\mathsf{H}})$ be a NIZK for a relation $\mathsf{CtMsgRel}^{\mathsf{sk}}$ defined by

$$\begin{aligned}\mathsf{CtMsgRel}^{\mathsf{sk}} = \{((\mathsf{ct}, c^{\mathsf{ee},j}, c_{\mathsf{ek}}), \omega \coloneqq (\mathsf{ek}, r, d^{\mathsf{ee}}, m, d_{\mathsf{ek}})) \ :& \\ \mathsf{ct} = \mathsf{Enc}^{\mathsf{cpa}}(\mathsf{ek}, m; r)& \\ \wedge \quad \mathsf{EECom.Verify}(\mathsf{pp}^{\mathsf{ee}}, c^{\mathsf{ee},j}, m, d_{\mathsf{ee}}) = 1& \\ \wedge \quad \mathsf{Com.Verify}(\mathsf{pp}^{\mathsf{com}}, c_{\mathsf{ek}}, \mathsf{ek}, d_{\mathsf{ek}}) = 1\}&\end{aligned}$$

that induces a langue $\mathscr{L}_{\mathsf{CtMsgRel}^{\mathsf{sk}}}$.

The obtained $\mathsf{MCFE}^{\mathsf{cca}}[\mathcal{F}, (\mathcal{Z}_{\lambda,i})_{i\in[n]}] = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$ is specified below

$\mathsf{Setup}(1^\lambda, 1^n)$**:** Given as input a security parameter λ, run $\mathsf{Setup}^{\mathsf{cpa}}(1^\lambda, 1^n)$ to obtain the master secret key $\mathsf{msk}^{\mathsf{cpa}}$ and encryption keys $\mathsf{ek}_i^{\mathsf{cpa}}$. Run $\mathsf{GenCRS}(1^\lambda)$ to obtain a structured reference string srs, sample $\mathsf{urs} \leftarrow \mathcal{URS}$ and set $\mathsf{crs} := (\mathsf{srs}, \mathsf{urs})$. Set up the commitment scheme $\mathsf{pp}^{\mathsf{com}} := \mathsf{Com.Setup}(1^\lambda)$, as well as $\mathsf{pp}^{\mathsf{ee}} := \mathsf{EECom.Setup}(1^\lambda)$. Commit to the encryption keys $(c_{\mathsf{ek},i}^{\mathsf{cpa}}, d_{\mathsf{ek},i}^{\mathsf{cpa}}) := \mathsf{Com.Commit}(\mathsf{pp}^{\mathsf{com}}, \mathsf{ek}_i^{\mathsf{cpa}}; r_{\mathsf{ek},i})$ with randomness $r_{\mathsf{ek},i} \xleftarrow{\$} \mathcal{C}_{\mathsf{rnd}}$, for each $i \in [n]$. Set $\mathsf{msk} := \mathsf{msk}^{\mathsf{cpa}}$

$$\mathsf{ek}_i := (\mathsf{ek}_i^{\mathsf{cpa}}, d_{\mathsf{ek},i}^{\mathsf{cpa}}, \mathsf{pk}_i, K_i)$$

where K_i is a PRF key. Set $\mathsf{pp} = (\lambda, n, \mathsf{crs}, \mathsf{pp}^{\mathsf{com}}, \mathsf{pp}^{\mathsf{ee}}, (c_{\mathsf{ek},i}^{\mathsf{cpa}})_i)$ and output $(\mathsf{pp}, \mathsf{msk}, (\mathsf{ek}_i)_i)$.

$\mathsf{Extract}(\mathsf{msk}, F_\lambda)$**:** Given a function description $F_\lambda : \prod_{i=1}^n (\mathcal{D}_{\lambda,i} \times \mathcal{Z}_{\lambda,i}) \to \mathcal{R}_\lambda$ in $\mathcal{F}$, and the master secret key msk, run $\mathsf{Extract}^{\mathsf{cpa}}(\mathsf{msk}^{\mathsf{cpa}}, F_\lambda)$ to obtain the decryption key dk_{F_λ}.

$\mathsf{Enc}(\mathsf{ek}_i, x_i, z_i)$**:** Given as inputs an encryption key $\mathsf{ek}_i = (\mathsf{ek}_i^{\mathsf{cpa}}, d_{\mathsf{ek},i}^{\mathsf{cpa}}, K_i)$,a message $x_i \in \mathcal{D}_{\lambda,i}$, and *public* input $z_i = (\mathsf{tag}, \tilde{z}_i) \in \mathcal{Z}_{\lambda,i}$ that contains some tag, compute[a]

$$\begin{aligned} r_{\mathsf{tag},i} &\leftarrow \mathsf{PRF}(K_i, x_i, \tilde{z}_i, \mathsf{tag}) \\ (c_{\mathsf{tag},i}^{\mathsf{ee},j}, d_{\mathsf{tag},i}^{\mathsf{ee}}) &\leftarrow \mathsf{EECom.Commit}(\mathsf{pp}^{\mathsf{ee}}, (x_i, z_i)) \\ \mathsf{ct}_{\mathsf{tag},i}^{\mathsf{cpa}} &\leftarrow \mathsf{Enc}^{\mathsf{cpa}}(\mathsf{ek}_i^{\mathsf{cpa}}, x_i, z_i; r_{\mathsf{tag},i}) \;\; . \end{aligned}$$

Use the NIZK to prove that,

$$((\mathsf{ct}_{\mathsf{tag},i}^{\mathsf{cpa}}, c_{\mathsf{tag},i}^{\mathsf{ee},j}, c_{\mathsf{ek},i}^{\mathsf{cpa}}), (\mathsf{ek}_i^{\mathsf{cpa}}, r_{\mathsf{tag},i}, d_{\mathsf{tag},i}^{\mathsf{ee}}, (x_i, z_i), d_{\mathsf{ek},i}^{\mathsf{cpa}}))$$

is a valid statement-witness pair for $\mathsf{CtMsgRel}^{\mathsf{sk}}$, where $c_{\mathsf{ek},i}^{\mathsf{cpa}}$ comes from pp. In the end, the proof is $\pi_{\mathsf{tag},i}$. Finally output the ciphertext $\mathsf{ct}_i := (\mathsf{ct}_i^{\mathsf{cpa}}, c_{\mathsf{tag},i}^{\mathsf{ee},j}, \pi_{\mathsf{tag},i})$.

$\mathsf{Dec}(\mathsf{dk}_{F_\lambda}, \mathbf{c})$**:** Given the decryption key dk_{F_λ} and a vector of ciphertexts $\mathbf{c}$ of length n, where for each $i \in [n]$ the i-th ciphertext is $\mathsf{ct}_i := (\mathsf{ct}_i^{\mathsf{cpa}}, c_{\mathsf{tag},i}^{\mathsf{ee},j}, \pi_{\mathsf{tag},i})$ run $\mathsf{Dec}^{\mathsf{cpa}}(\mathsf{dk}_{F_\lambda}, \mathbf{c}^{\mathsf{cpa}} := (\mathsf{ct}_i^{\mathsf{cpa}})_i)$ to obtain $F(x)$.
If for some i the NIZK verifies fail $\mathsf{Verify}^{\mathsf{H}}(\mathsf{crs}, (\mathsf{ct}_{\mathsf{tag},i}^{\mathsf{cpa}}, c_{\mathsf{tag},i}^{\mathsf{ee},j}, c_{\mathsf{ek},i}^{\mathsf{cpa}}), \pi_{\mathsf{tag},i}) = 0$, then output $\perp$. Otherwise output $F(x)$.

[a] The $\mathsf{Enc}^{\mathsf{cpa}}$ is probabilistic as we make the hypothesis in our security proof later that $\mathsf{MCFE}^{\mathsf{cpa}}[\mathcal{F}, (\mathcal{Z}_{\lambda,i})_{i\in[n]}]$ is IND-CPA secure. The randomness $r_{\mathsf{tag},i}$ is used as parameters for the sampling during encryption, differing for each repetition x_i, z_i on tag. One can of course make $r_{\mathsf{tag},i}$ implicit in the probabilistic run of $\mathsf{Enc}^{\mathsf{cpa}}$.

Correctness. Following the *correctness* of the underlying MCFE and NIZK, as well as the *correctness* of the commitment Com, the construction is correct.

Security. The following theorem proves the CCA-security, under *pos*-restriction *i.e.* for any $(\mathsf{tag}, \tilde{z})$ that is queried to **LoR** or **Enc**, the adversary $\mathcal{A}$ either queries all $i \in \mathcal{H}$ or none, and *static* corruption. For better clarity of the proof's structure, we consider the *one-challenge* setting: there is only one challenge tag tag^* that is asked to **LoR**.

Theorem 33. *If the* admissibility *of $\mathcal{F}$ is efficiently decidable and: $\mathsf{MCFE}^{\mathsf{cpa}}[\mathcal{F}, (\mathcal{Z}_{\lambda,i})_{i\in[n]}]$ is* IND-CPA *secure;* Com *is a* perfectly binding *and* computationally hiding *commitment scheme; PRF is a secure PRF;* NIZK *is a NIZK for the relation* $\mathsf{CtMsgRel}^{\mathsf{sk}}$*, satisfying* correctness*,* zero-knowledge*, and* simulation soundness*;* EECom *is an* equivocable-extractable *commitment scheme; then $\mathsf{MCFE}^{\mathsf{cca}}[\mathcal{F}, (\mathcal{Z}_{\lambda,i})_{i\in[n]}]$ is* pos-statically *IND-CCA secure.*

Full proof is provided in Appendix G and explanations for our design choices are given in Section 1.2.

4.2 CCA-Security for Inner Products

In this section we give concrete instantiation of the multi-client CCA-secure IPFE construction following Section 4, using different building blocks.

The underlying IND-CPA MCFE. The scheme $\mathsf{MCFE}^{\mathsf{cpa}}$ is depicted in Figure 6 in Appendix B. In order to highlight the instantiation of our generic transformation from Section 4.1, we adapt the MCFE of [67] to the simpler case of inner products. Our transformation from Theorem 33 works for general function classes that have public inputs, therefore will cover the original MCFE of [67] when being plugged in with other components (*i.e.* the NIZK and the commitment schemes). The more general instantiation to cover the original MCFE of [67] is given in the full version of this work. The *correctness* and *CPA-security* follow [67].

Theorem 34 (Corollary 13, [67]). *Let $\mathsf{MCFE}^{\mathsf{cpa}}$ be a MCFE scheme with fine-grained access control for the function class $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$, given in Figure 6 in a bilinear group setting $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{t}, g_1, g_2, g_\mathsf{t}, \mathbf{e}, q)$. Then, in the random oracle model, $\mathcal{E}$ is* statically IND-secure *against* complete *challenge queries(*as per *Definition 30 and its weaker notions), under the* SXDH *in* $\mathbb{G}_1$ *and* $\mathbb{G}_2$.

Recall: The Groth-Sahai NIZK. We use the Groth-Sahai NIZK [49] in the bilinear group setting under SXDH, inparticular its commitment step to commit to the encryption keys in $(c^{\mathsf{cpa}}_{\mathsf{ek},i}, d^{\mathsf{cpa}}_{\mathsf{ek},i}) \coloneqq \mathsf{Com.Commit}(\mathsf{pp}^{\mathsf{com}}, \mathsf{ek}^{\mathsf{cpa}}_i; r_{\mathsf{ek},i})$ of the public parameters. Later on, in the common NIZK, we employ later steps of Groth-Sahai to prove the correctness of the encryption keys. We use prime-order bilinear group setting $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{t}, g_1, g_2, g_\mathsf{t}, \mathbf{e}, q)$[15]. Details are recalled in Appendix E.

The NIZK proof systems. We instantiate the $\mathsf{NIZK} = (\mathsf{GenCRS}, \mathsf{Prove}^{\mathsf{H}}, \mathsf{Verify}^{\mathsf{H}})$ for relation $\mathsf{CtMsgRel}^{\mathsf{sk}}_{\mathsf{ip}}$ defined by

$$\begin{aligned}\mathsf{CtMsgRel}^{\mathsf{sk}} = \{((\mathsf{ct}, c^{\mathsf{ee},j}, c_{\mathsf{ek}}),\quad \omega \coloneqq (\mathsf{ek}, r, d^{\mathsf{ee}}, m, d_{\mathsf{ek}}))\ :\\ \mathsf{ct} = \mathsf{Enc}^{\mathsf{cpa}}(\mathsf{ek}, m; r)\quad \wedge\quad \mathsf{EECom.Verify}(\mathsf{pp}^{\mathsf{ee}}, c^{\mathsf{ee},j}, m, d_{\mathsf{ee}}) = 1\quad \wedge\quad \mathsf{Com.Verify}(\mathsf{pp}^{\mathsf{com}}, c_{\mathsf{ek}}, \mathsf{ek}, d_{\mathsf{ek}}) = 1\}\ .\end{aligned}$$

We describe in details the statements and their witnesses in Appendix D. We recall that in the context of our MCFE, the values $[\![(\omega, \omega')]\!]_1$ come from a hash of tag $\mathsf{H}(\mathsf{tag})$ and can be publicly computed. The same goes for the multi-Cramer-Shoup hash $\sigma = \mathsf{H}^{\mathsf{cr}}(\mathsf{tag}, (u_{i,j}, v_{i,j}, e_{i,j})_{i,j})$ that is needed for the extractable-equivocable commitment (Fig. 1). The plan to construct the NIZK proof system is as follows:

Step 1: We first give a Σ-protocol $\Sigma_{\mathsf{cpa,ee}}$ for proving the parts that relate to the $\mathsf{MCFE}^{\mathsf{cpa}}$ and EECom. The Σ-protocol is executed by each client $i \in [n]$.
Step 2: We then apply the Fiat-Shamir transform to the Σ-protocol from **Step 1** to obtain a non-interactive proof system $\Pi^{\mathsf{ss}}_{\mathsf{cpa,ee}}$. Thanks to the 2-special soundness of the Σ-protocol, applying then the result from [40] provides simulation-soundness for $\Pi^{\mathsf{ss}}_{\mathsf{cpa,ee}}$.
Step 3: We finally apply the Groth-Sahai NIZK, after enhancing with techniques from [48, 24, 50, 51] to achieve Groth-Sahai NIZK $\Pi^{\mathsf{ss}}_{\mathsf{gs}}$ with simulation-soundness. The final commitment $c_{\mathsf{ek},i}$ of encryption keys is done by the Grothh-Sahai-induced commitment in $\Pi^{\mathsf{ss}}_{\mathsf{gs}}$. Then the NIZK $\Pi^{\mathsf{ss}}_{\mathsf{gs}}$ is used to prove the remaining correctness of the encryption keys, involving the multi-scalar equations in (23) that appear during encryption to ct_i. This Groth-Sahai part is concatenated into the NIZK proof system from **Step 2**.

The general overview is in Section 1.2 and full details can be found in Appendix C. We conclude this section on instantiations for IND-CCA secure MCFE for $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$ by the following corollary.

Corollary 35. *Assuming the* SXDH *and* DLIN *assumptions in a bilinear group setting in the ROM, there exists an IND-CCA secure MCFE for computing sum of inner products following* $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$, *with* adaptive challenge *security,* static *corruption, and* pos-*security,* as per *Definition 30.*

[15] We writing the groups additively.

5 CCA-Security for Multi-Input FE for Inner Products: Constructions

We turn our attention to building IND-CCA secure MIFE for function classes with auxiliary inputs, with respect to Definition 30 but in the case of MIFE: a MIFE can be seen as a MCFE with security *as per* Definition 30 with one tag being fixed for all encryption. In other words, the CCA-secure MCFE from previous section can be turnen MIFE by fixing a public tag. We present here another approach that is more competitive in terms of efficiency based on SPHF. Formally, the syntax and definition of IND-CPA security for MIFE are given in Definitions 45 as well as 46 of Appendix A.4. The definition of IND-CCA security for MIFE is given in Definition 47 of Appendix A.4.

The roadmap towards our IND-CCA secure MIFE for inner products from SPHF (see preliminaries in Section 2.2) is as follows:

Step 1: We start by giving an IND-CPA secure MIFE from SPHF, for $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$ (see Definition 27), is given in Figure 4. We verifiy that the construction is correct and then prove its security.

Step 2: We then add a layer of *tag-based* SPHF to the construction in Figure 4 to achieve IND-CCA security. This resembles what is done for the case of IND-CCA secure *public-key functional encryption* in [18].

Step 3: Finally we discuss how to instantiate all the building blocks of the construction, both the SPHF for the IND-CPA secure MIFE and the *tag-based* SPHF to get IND-CCA security.

5.1 Generic Constructions from SPHF

This section presents **Step 1** of our roadmap to construct an IND-CPA secure MIFE from SPHF for the function class $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$. The construction is given in Figure 4, and a full proof is given in Theorem 52. This IND-CPA secure MIFE plays an important role in the next step to achieve IND-CCA security in **Step 2**.

Step 1: IND-CPA Secure MIFE from SPHF. Let

$$\mathsf{PHF} = (\mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$$

be a SPHF for the subset membership problem $\mathbf{P}$, having hash key space $\mathcal{K}$ and projected key space $\mathcal{K}^{\mathsf{pj}}$. Let $\mathsf{PRF} = (\mathsf{Keygen}, \mathsf{Eval})$ be a PRF with key space $\mathcal{K}^{\mathsf{prf}}$, domain $\mathcal{D}^{\mathsf{prf}}$, and range $\mathcal{R}^{\mathsf{prf}}$. We use a group setting $\mathbb{G}$ that contains the hash values of the PHF, writtent additively. We suppose that PHF is $(\mathsf{hk}_\perp, g_\perp, M_\perp, M_z, \epsilon_{\mathrm{ti}})$-CPA-friendly (following Definition 12). For the ease of presenting the construction, we make the following assumptions:

1. the group element $g_\perp$ is a generator of $\mathbb{G}$, its order is $M_\perp = |\mathbb{G}| = q$ prime, and in the following the implicit notation $[\![\cdot]\!]$ is with respect to $g_\perp$,
2. the integer M_{ti} corresponds to the range bound in $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$,
3. there is a ppt algorithm that, when given $[\![\sum_{i=1}^n \langle \mathbf{x}_i, \mathbf{y}_i\rangle]\!]$, can find the discrete log in base $g_\perp$ and output $\sum_{i=1}^n \langle \mathbf{x}_i, \mathbf{y}_i\rangle$.

Figure 4 gives the construction of an IND-CPA secure MIFE from PHF for $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$. Full proof for security and correctness are in Appendix H.

$\mathsf{Setup}(1^\lambda, 1^n)$**:** Given as input a security parameter λ and a number of slots n. Sample $\Lambda \overset{\$}{\leftarrow} I_\lambda$ of $\mathbf{P}$. Sample a random statement-witness $(\mathbf{b}, \omega)$ relative to Λ, where $\mathbf{b} \in \mathcal{X}$ is a yes-instance. For $i \in [n]$, sample $K \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$ for the PRF. Then run

$$\forall i \in [n], k \in [N] : \mathsf{hk}_{i,k} \leftarrow \mathsf{hashkg}(\Lambda) \ ; \ \mathsf{pjk}_{i,k} \leftarrow \mathsf{projkg}(\mathsf{hk}_{i,k}, \mathbf{b}) \ .$$

Then denote the public parameter $\mathsf{pp} = (\lambda, n, \mathbf{b}, \Lambda)$. Finally, define $\mathsf{hk}_i \overset{def}{=} (\mathsf{hk}_{i,k})_{k\in[N]}$, $\mathsf{pjk}_i \overset{def}{=} (\mathsf{pjk}_{i,k})_{k\in[N]}$ output

$$\mathsf{msk} \overset{def}{=} (\mathsf{hk}_i)_{i=1}^n; \quad \forall\, i \in [n] : \mathsf{ek}_i \overset{def}{=} (K, \mathsf{pjk}_i, \mathsf{hk}_i, \mathbf{b}, \omega) \ .$$

$\mathsf{Extract}(\mathsf{msk}, F_\lambda)$**:** Given a function description F_λ in $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$, with parameters $(\mathbf{y}_i)_{i=1}^n$, and the master secret key msk, for each $i \in [n]$, sample random coefficients $\delta^{\mathsf{ss}}_{j,i} \overset{\$}{\leftarrow} \mathbb{Z}_q$, where $j \in [n]$, as well as hash keys $\mathsf{hk}^{\mathsf{ss}}_i \in \mathcal{K}$ that satisfy

$$\langle \mathbf{y}_i, \mathsf{hk}_i \rangle + \sum_{j\in[n]} \delta^{\mathsf{ss}}_{j,i} \cdot \langle \mathbf{y}_i, \mathsf{hk}_j \rangle = 0 \ \text{ and } \ \sum_{i\in[n]} \langle \mathbf{y}_i, \mathsf{hk}^{\mathsf{ss}}_i \rangle = 0 \ . \tag{8}$$

Compute, for all $i \in [n]$, $\mathsf{HK}[j, i] \overset{def}{=} \delta^{\mathsf{ss}}_{j,i} \cdot \langle \mathbf{y}_i, \mathsf{hk}_j \rangle \in \mathcal{K}$, then define $\mathsf{HK}^{\mathsf{ss}}[i] = \langle \mathbf{y}_i, \mathsf{hk}_i + \mathsf{hk}^{\mathsf{ss}}_i \rangle$, and output $\mathsf{dk}_F \overset{def}{=} (\mathsf{HK}, \mathsf{HK}^{\mathsf{ss}})$.

$\mathsf{Enc}(\mathsf{ek}_i, \mathbf{x}_i)$**:** Parse the encryption key $\mathsf{ek}_i = (K, \mathsf{pjk}_i, \mathsf{hk}_i, \mathbf{b}, \omega)$, a message $\mathbf{x}_i \in \mathcal{D}_{\lambda,i}$. Compute

$$\begin{aligned} r_i &\leftarrow \mathsf{PRF}(K, \mathbf{x}_i); \\ \mathbf{b}_i &\leftarrow \mathsf{RSR}(\lambda, \mathbf{b}, r_i); \quad \omega_i \leftarrow \mathsf{RSRw}(\lambda, \mathbf{b}, r_i, \omega) \\ \mathsf{pjk}^{\mathsf{Enc}}_{i,k} &\leftarrow \mathsf{projkg}(\mathsf{hk}_{i,k}, \mathbf{b}_i) \\ \text{for } k \in [N] : \mathbf{c}_{i,k} &\overset{def}{=} \mathsf{projhash}(\mathsf{pjk}^{\mathsf{Enc}}_{i,k}, \mathbf{b}_i, \omega_i) + \mathsf{projhash}(\mathsf{pjk}_{i,k}, \mathbf{b}, \omega) + [\![\mathbf{x}_{i,k}]\!] \in \mathbb{G} \ . \end{aligned}$$

Define $\mathbf{c}_i \overset{def}{=} (\mathbf{c}_{i,k})_{k=1}^N \in \mathbb{G}^N$ and output $\mathsf{ct}_i := (\mathbf{c}_i, \mathbf{b}_i)$.

$\mathsf{Dec}(\mathsf{dk}_F, (\mathsf{ct}_i)_{i=1}^n)$**:** Given the decryption key dk_{F_λ} and a vector of ciphertexts $\mathsf{ct}_i \overset{def}{=} (\mathbf{c}_i, \mathbf{b}_i)_i$ of length n, parse $\mathsf{dk}_F \overset{def}{=} (\mathsf{HK}, \mathsf{HK}^{\mathsf{ss}})$. For each $i \in [n]$, compute, where $\mathbf{b}$ comes from pp,

$$\begin{aligned} \mathsf{pt}_i &\overset{def}{=} \left(\sum_{k\in[N]} \mathbf{y}_{i,k} \cdot \mathbf{c}_{i,k} \right) - \mathsf{hash}(\mathsf{HK}^{\mathsf{ss}}[i], \mathbf{b}) \\ \forall j \in [n] : \mathsf{share}[j, i] &\overset{def}{=} \mathsf{hash}(\mathsf{HK}[j, i], \mathbf{b}_i) \end{aligned}$$

and $\mathsf{out} \overset{def}{=} \sum_{i=1}^n \left(\mathsf{pt}_i + \sum_{j\in[n]} \mathsf{share}[j, i]\right) - \mathsf{hash}(0, \mathbf{b}) - \sum_{i=1}^n \mathsf{hash}(0, \mathbf{b}_i)$. Finally, find the discrete log in base $g_\perp$ of out.

Fig. 4: The IND-CPA MIFE from SPHF for $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$ (see Definition 27). The algorithms RSR and RSRw are the self-reduction algorithms for the subset membership problem $\mathbf{P}$, we recall in the preliminaries in Section 2.2.

$\mathsf{Setup}(1^\lambda, 1^n)$**:** Given as input a security parameter λ and a number of slots n. Sample $\Lambda \xleftarrow{\$} I_\lambda$ of $\mathbf{P}$. Sample a random statement-witness $(\mathbf{b}, \omega)$ relative to Λ, where $\mathbf{b} \in \mathcal{X}$ is a yes-instance. For $i \in [n]$, sample $K \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$ for the PRF. Then run

$$\forall i \in [n], k \in [N] : \mathsf{hk}_{i,k} \leftarrow \mathsf{hashkg}(\Lambda) \ ; \ \mathsf{pjk}_{i,k} \leftarrow \mathsf{projkg}(\mathsf{hk}_{i,k}, \mathbf{b})$$
$$\boxed{\mathsf{hk}_i^{\mathsf{tb}} \leftarrow \mathsf{hashkg}^{\mathsf{tb}}(\Lambda)} \ ; \ \boxed{\mathsf{pjk}_i^{\mathsf{tb}} \leftarrow \mathsf{projkg}^{\mathsf{tb}}(\mathsf{hk}_i^{\mathsf{tb}})} \ .$$

Then publish the public parameter $\mathsf{pp} = (\lambda, n, \mathbf{b}, \Lambda)$ and define $\mathsf{hk}_i \stackrel{def}{=} (\mathsf{hk}_{i,k})_{k \in [N]}, \mathsf{pjk}_i \stackrel{def}{=} (\mathsf{pjk}_{i,k})_{k \in [N]}$. Finally, output

$$\mathsf{msk} \stackrel{def}{=} (\mathsf{hk}_i, \boxed{\mathsf{hk}_i^{\mathsf{tb}}})_{i=1}^n; \quad \forall\, i \in [n] : \mathsf{ek}_i \stackrel{def}{=} (K, \mathsf{pjk}_i, \boxed{\mathsf{pjk}_i^{\mathsf{tb}}}, \mathbf{b}, \omega) \ .$$

$\mathsf{Extract}(\mathsf{msk}, F_\lambda)$**:** Given a function description F_λ in $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$, with parameters $(\mathbf{y}_i)_{i=1}^n$, and the master secret key msk, for each $i \in [n]$, sample random coefficients $\delta_{j,i}^{\mathsf{tb,ss}}, \delta_{j,i}^{\mathsf{ss}} \xleftarrow{\$} \mathbb{Z}_q$, where $j \in [n]$, as well as hash keys $\mathsf{hk}_i^{\mathsf{tb,ss}}, \mathsf{hk}_i^{\mathsf{ss}} \in \mathcal{K}$ that satisfy

$$\langle \mathbf{y}_i, \mathsf{hk}_i \rangle + \sum_{j \in [n]} \delta_{j,i}^{\mathsf{ss}} \cdot \langle \mathbf{y}_i, \mathsf{hk}_j \rangle = 0 \ \text{ and } \ \sum_{i \in [n]} \langle \mathbf{y}_i, \mathsf{hk}_i^{\mathsf{ss}} \rangle = 0 \tag{9}$$

$$\langle \mathbf{y}_i, \mathsf{hk}_i \rangle + \sum_{j \in [n]} \boxed{\delta_{j,i}^{\mathsf{tb,ss}}} \cdot \langle \mathbf{y}_i, \mathsf{hk}_j^{\mathsf{tb}} \rangle = 0 \ \text{ and } \ \sum_{i \in [n]} \langle \mathbf{y}_i, \boxed{\mathsf{hk}_i^{\mathsf{tb,ss}}} \rangle = 0 \ . \tag{10}$$

Compute, for all $i \in [n]$, $\mathsf{HK}[j,i] \stackrel{def}{=} \delta_{j,i}^{\mathsf{ss}} \cdot \langle \mathbf{y}_i, \mathsf{hk}_j \rangle \in \mathcal{K}$, then define $\mathsf{HK}^{\mathsf{ss}}[i] = \langle \mathbf{y}_i, \mathsf{hk}_i + \mathsf{hk}_i^{\mathsf{ss}} \rangle$. Compute, for all $i \in [n]$, $\boxed{\mathsf{HK}^{\mathsf{tb}}[j,i]} \stackrel{def}{=} \delta_{j,i}^{\mathsf{tb,ss}} \cdot \langle \mathbf{y}_i, \mathsf{hk}_j^{\mathsf{tb}} \rangle \in \mathcal{K}$, then define $\boxed{\mathsf{HK}^{\mathsf{tb,ss}}[i]} = \langle \mathbf{y}_i, \mathsf{hk}_i^{\mathsf{tb}} + \mathsf{hk}_i^{\mathsf{tb,ss}} \rangle$. Finally, output $\mathsf{dk}_F \stackrel{def}{=} (\mathsf{HK}, \mathsf{HK}^{\mathsf{ss}}, \boxed{\mathsf{HK}^{\mathsf{tb}}, \mathsf{HK}^{\mathsf{tb,ss}}})$.

$\mathsf{Enc}(\mathsf{ek}_i, \mathbf{x}_i, \boxed{\mathsf{tag}})$**:** Parse the encryption key $\mathsf{ek}_i = (K, \mathsf{pjk}_i, \mathbf{b}, \omega)$, a message $\mathbf{x}_i \in \mathcal{D}_{\lambda,i}$. Compute

$$r_i \leftarrow \mathsf{PRF}(K, \mathbf{x}_i);$$
$$\mathbf{b}_i \leftarrow \mathsf{RSR}(\lambda, \mathbf{b}, r_i); \quad \omega_i \leftarrow \mathsf{RSRw}(\lambda, \mathbf{b}, r_i, \omega)$$
$$\text{for } k \in [N] : \mathbf{c}_{i,k} \stackrel{def}{=} \mathsf{projhash}(\mathsf{pjk}_{i,k}, \mathbf{b}_i, \omega_i) + \mathsf{projhash}(\mathsf{pjk}_{i,k}, \mathbf{b}, \omega) + [\![\mathbf{x}_{i,k}]\!] \in \mathbb{G}$$
$$\text{for } k \in [N] : \boxed{\mathbf{c}_{i,k}^{\mathsf{tb}}} \stackrel{def}{=} \mathsf{projhash}^{\mathsf{tb}}(\mathsf{pjk}_{i,k}^{\mathsf{tb}}, \mathbf{b}_i, \omega_i, \mathsf{tag}) + \mathsf{projhash}^{\mathsf{tb}}(\mathsf{pjk}_{i,k}, \mathbf{b}, \omega, \mathsf{tag}) \in \mathbb{G} \ .$$

Define $\mathbf{c}_i \stackrel{def}{=} (\mathbf{c}_{i,k}, \boxed{\mathbf{c}_{i,k}^{\mathsf{tb}}})_{k=1}^N \in \mathbb{G}^N$ and output $\mathsf{ct}_i := (\mathbf{c}_i, \mathbf{b}_i)$.

$\mathsf{Dec}(\mathsf{dk}_F, (\mathsf{ct}_i)_{i=1}^n, \boxed{\mathsf{tag}})$**:** Given the decryption key dk_{F_λ} and a vector of ciphertexts $\mathsf{ct}_i \stackrel{def}{=} (\mathbf{c}_i, \mathbf{b}_i)_i$ of length n, parse $\mathsf{dk}_F \stackrel{def}{=} (\mathsf{HK}, \mathsf{HK}^{\mathsf{ss}})$. For each $i \in [n]$, compute, where $\mathbf{b}$ comes from pp,

$$\mathsf{pt}_i \stackrel{def}{=} \left(\sum_{k \in [N]} \mathbf{y}_{i,k} \cdot \mathbf{c}_{i,k} \right) - \mathsf{hash}(\mathsf{HK}^{\mathsf{ss}}[i], \mathbf{b}); \qquad \boxed{\mathsf{pt}_i^{\mathsf{tb}}} \stackrel{def}{=} \left(\sum_{k \in [N]} \mathbf{y}_{i,k} \cdot \mathbf{c}_{i,k}^{\mathsf{tb}} \right) - \mathsf{hash}^{\mathsf{tb}}(\mathsf{HK}^{\mathsf{tb,ss}}[i], \mathbf{b}, \boxed{\mathsf{tag}})$$
$$\forall j \in [n] :$$
$$\mathsf{share}[j,i] \stackrel{def}{=} \mathsf{hash}(\mathsf{HK}[j,i], \mathbf{b}_i); \qquad \boxed{\mathsf{share}^{\mathsf{tb}}[j,i]} \stackrel{def}{=} \mathsf{hash}^{\mathsf{tb}}(\mathsf{HK}^{\mathsf{tb}}[j,i], \mathbf{b}_i, \boxed{\mathsf{tag}}) \ .$$

Check $\sum_{i=1}^n \left(\mathsf{pt}_i^{\mathsf{tb}} + \sum_{j \in [n]} \mathsf{share}^{\mathsf{tb}}[j,i] \right) - \mathsf{hash}^{\mathsf{tb}}(0, \mathbf{b}, \boxed{\mathsf{tag}}) - \sum_{i=1}^n \mathsf{hash}^{\mathsf{tb}}(0, \mathbf{b}_i, \boxed{\mathsf{tag}}) \stackrel{?}{=} 0$. If true compute out $\stackrel{def}{=} \sum_{i=1}^n \left(\mathsf{pt}_i + \sum_{j \in [n]} \mathsf{share}[j,i] \right) - \mathsf{hash}(0, \mathbf{b}) - \sum_{i=1}^n \mathsf{hash}(0, \mathbf{b}_i)$ and output the discrete log in base $g_\perp$ of out. Otherwise output $\perp$.

Fig. 5: The $\boxed{\text{tag-based}}$ IND-CCA secure MIFE from SPHF and $\boxed{\text{tag-based}}$ SPHF for $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$ (see Definition 27).

Step 2: IND-CCA Secure MIFE from tag-based SPHF. We need a further property from the SPHF PHF to achieve IND-CCA security, then we add another layer of *tag-based* SPHF to the construction in Figure 4 to achieve IND-CCA security. In particular, we require a form of *CCA-friendliness* from a tag-based SPHF tPHF to achieve IND-CCA security. The syntax of tPHF and CCA-friendly properties are recalled in Definition 13 together with Definition 17.

We let PHF = (hashkg, projkg, hash, projhash) be a SPHF for the subset membership problem $\mathbf{P}$. Let PRF = (Keygen, Eval) be a PRF with key space $\mathcal{K}$, domain $\mathcal{D}$, and range $\mathcal{R}$. We suppose that PHF is $(\mathsf{hk}_\perp, g_\perp, M_\perp, M_z, \epsilon_{\mathsf{ti}})$-CPA-friendly (following Definition 12). We need a tag-based SPHF tPHF = $(\mathsf{hashkg}^{\mathsf{tb}}, \mathsf{projkg}^{\mathsf{tb}}, \mathsf{hash}^{\mathsf{tb}}, \mathsf{projhash}^{\mathsf{tb}})$ that is $(\mathsf{hashkg}^{\mathsf{tb}}, \mathsf{Coeff}, \epsilon_{2\mathsf{u}}^{\mathsf{tb}}, M_z, \epsilon_{\mathsf{ti}}^{\mathsf{tb}})$-CCA-friendly (following Definition 17). Both PHF and tPHF are defined over the same subset membership problem $\mathbf{P}$. The group setting $\mathbb{G}$ is used such that it contains the hash values of the PHF and tPHF, written additively. Moreover, we assume the conventions in Items 1 to 3.

The main blueprint to obtain an IND-CCA secure MIFE is as follows:

Step 2a: We first construct a *tag-based* MIFE. The encryption and decryption algorithms receive a tag tag. The corresponding *tag-based* IND-CCA security then ensure that as long as the challenge tag tag^* is not queried to the decryption oracle, the adversary cannot distinguish the challenge ciphertexts. The tag-based tPHF is used to process this tag. The construction is given in Figure 5.
Step 2b: As in classical treatments of IND-CCA security, we then use a *one-time* signature and a collision hash function to turn the above tag-based MIFE into an IND-CCA secure MIFE: fresh verification keys of the one-time signature are hashed to obtain as tags for encryptions. At a high level, the collision-resistance of the hash function ensures fresh ciphertexts will come with fresh tags, and this case is covered by the tag-based IND-CCA security. Otherwise, the adversary somehow manages to query the decryption oracle with the challenge tag on some ciphertext of their choice, but this case leads to a forgery of the one-time signature.

The following focuses on the more technical part of **Step 2a**, *i.e.* the construction of a tag-based MIFE. Figure 5 gives the construction of a tag-based IND-CCA secure MIFE with *tags* from PHF and tPHF for $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$. The *correctness* is adapted to be quantified over any $\mathsf{tag} \in \mathsf{Tag}$ furthermore. The *security* model is a tag-based IND-CCA security notion, where the principal differences are:

- For each query to the encryption and decryption oracles, the adversary must provide a tag tag.
- After querying the challenge ciphertext on some challenge tag tag^*, in the **Finalize** procedure, it is checked that the tag tag^* is not queried to the decryption oracle.

For completeness, formal details are given in Definition 48. We boostrap our construction from the IND-CPA secure MIFE in Figure 4 using the tag-based tPHF, main differences are $\boxed{\text{boxed}}$.

We formalize the tag-based IND-CCA security in Theorem 36. We provide an informal sketch of main points as follows. Full details will be put in the full version of this work.

Theorem 36. *Let* $\mathsf{PHF} = (\mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$ *be a SPHF for the subset membership problem* $\mathbf{P}$*,* as per *Definition 5. We also use a* tag-based *SPHF for the same* $\mathbf{P}$*, namely* $\mathsf{tPHF} = (\mathsf{hashkg}^{tb}, \mathsf{projkg}^{tb}, \mathsf{hash}^{tb}, \mathsf{projhash}^{tb})$*. We suppose further that* $\mathbf{P}$ *is hard* $\overset{\frown}{\text{and}}$ *average self-reducible, PHF is CPA-friendly (following Definition 12), and tPHF is* $(\mathsf{hashkg}^{tb}, \mathit{Coeff}, \epsilon^{tb}_{2\mathrm{u}}, M_z, \epsilon^{tb}_{\mathrm{ti}})$*-CCA-friendly (following Definition 17) . Then the MIFE in Figure 5 for* $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$ *is tag-based IND-CPA* as per *Definition 48.*

Proof (Sketch, for Theorem 36). We highlight important details below:

- The MIFE in Figure 5 is *tag-based*: the encryption and decryption receives a tag tag.
- Our main usage of the tag-based SPHF tPHF is in the spirit of proving consistency (for achieving non-malleability) of *each* ciphertext component $\mathbf{c}_{i,k}$ for $k \in [N]$ of each slot i.
- In particular, the tag-based hash values $\boxed{\mathbf{c}^{\mathsf{tb}}_{i,k}}$ are computed on the same instances $\mathbf{b}_i, \mathbf{b}$ that are used for encrypting $\mathbf{x}_i = (\mathbf{x}_{i,k})_{k\in[N]}$.
- Later in decryption, a check on a linear relation of the tag-based hash values $\mathbf{c}^{\mathsf{tb}}_{i,k}$ is performed to ensure consistency:

$$\sum_{i=1}^{n}\left(\mathsf{pt}^{\mathsf{tb}}_i + \sum_{j\in[n]} \mathsf{share}^{\mathsf{tb}}[j,i]\right) - \mathsf{hash}^{\mathsf{tb}}(0, \mathbf{b}, \boxed{\mathsf{tag}}) - \sum_{i=1}^{n} \mathsf{hash}^{\mathsf{tb}}(0, \mathbf{b}_i, \boxed{\mathsf{tag}}) \overset{?}{=} 0$$

wher $\mathsf{share}^{\mathsf{tb}}[j,i], \mathsf{pt}^{\mathsf{tb}}_i$ are computed from the tag-based hash values $\mathbf{c}^{\mathsf{tb}}_{i,k}$ and the tag-based hash keys from the functional key dk_F for inner products with parameters $(\mathbf{y}_i)_i$. At an intuition level, thanks to the CCA-friendliness of tPHF, this makes sure that even with the information on the hash keys from dk_F, conditioned on the received ciphertexts, the adversary cannot forge a new ciphertext with $\mathbf{c}^{\mathsf{tb}}_{i,k}$'s that pass the checks but are not consistent with $\mathbf{c}_{i,k}$. For instance, this prevents malleating and giving $\widehat{\mathbf{c}}_{i,k}$ whose linear combination *as per* decryption does not give the correct inner products[16].

- Technically, the CCA-friendliness intervenes when we switch the instances in the encryption process, from a yes-instance $\mathbf{b}$ to a no-instance $\widehat{\mathbf{b}}$, then change the challenge ciphertexts $\mathbf{c}^{(b,j)}_{i,k}$ to be independent from the challenge bit b.
  - The hardness of the subset membership problem $\mathbf{P}$ is used to switch from yes-instances to no-instances.
  - The no-instances are embedded since set up time, for hash key and projection key generation. The key simulation is ensured by the CCA-friendliness of tPHF and CPA-friendliness of PHF, particularly the *Universal Translation Indistinguishability* property of tPHF (following Definition 16) and the *Translation Indistinguishability* property of PHF (following Definition 11).
  - The 2-universality of tPHF (following Definition 15), given that no-instances are used for encryption, ensures statistically that without the hash key, the adversary cannot forge decryption queries in which they submit a malleated challenge ciphertext $\widetilde{\mathbf{c}^{(b,j)}_{i,k}}$ that is associated by tag-based components $\widetilde{\mathbf{c}^{\mathsf{tb}}_{i,k}}$ that verify.
  - Notably, the latter means the foregoing decryption query on $\widetilde{\mathbf{c}^{(b,j)}_{i,k}}$ has a different tag' (constrained by the tag-based IND-CCA) and in $\widetilde{\mathbf{c}^{\mathsf{tb}}_{i,k}}$ there is a fresh hash value on a no-instance (from the public parameters that we embed in the setup) and tag'. The 2-universality of tPHF ensures this happens up to a $\epsilon^{\mathsf{tb}}_{2\mathrm{u}}$-multiplicative factor of the probability we simulate the given $\mathbf{c}^{(b,j)}_{i,k}$, which is negligible when $\epsilon^{\mathsf{tb}}_{2\mathrm{u}}$ is polynomially large, thanks to the smoothness of tPHF over hashes on no-instances.
  - The simulation of procedures **Setup, KeyGen, Enc** closely follow what is done in game G_5 of the proof of IND-CPA security.

In the end, when the challenge ciphertexts $\mathbf{c}^{(b,j)}_{i,k}$ are independent from the challenge bit b, the advantage of the adversary is zero, and the proof is concluded. □

[16] It is clear that we do not protect against valid linear combinations of ciphertexts basing on information from a functional key, for decryption queries, but this is not a problem due to the linearity of inner products and the fact that the adversary possesses the functional key themselves.

5.2 Instantiations

From DDH. The $\mathsf{PHF} = (\mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$ that meets the requirements of our transformation in Theorem 36 can be instantiated by the Gennaro-Lindell SPHF introduced in [43, Sect 3.2], which can be instantiated from DDH (similar to the Cramer-Shoup SPHF to obtain CCA2-security in [34, 35]). Lemma 37 shows that this SPHF satisfies the key-homomorphic property, and more generally, is FE-CPA-friendly. This a corollary of [18, Lemma 13] that proves CPA-friendliness for the SPHF from [34, 35] under DDH.

Lemma 37 (Corollary of Lemma 13 of [18]). *The Gennaro-Lindell SPHF from [43] satisfies the key-homomorphic property, and is FE-CPA-friendly.*

The $\mathsf{tPHF} = (\mathsf{hashkg}^{\mathsf{tb}}, \mathsf{projkg}^{\mathsf{tb}}, \mathsf{hash}^{\mathsf{tb}}, \mathsf{projhash}^{\mathsf{tb}})$ that meets the requirements of our transformation in Theorem 36 can be instantiated by [35, 4] based on DDH. Its proof for FE-CCA-friendliness can be found in [18, Lemma 27].

From DCR. The $\mathsf{PHF} = (\mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$ that meets the requirements of our transformation in Theorem 36 can be also instantiated from N-residuosity[17] as in [43, Sect 8.5]. Lemma 38 shows that this SPHF satisfies the key-homomorphic property, and more generally, is FE-CPA-friendly. Specifically, the SPHF in [43, Sect 8.5] has the structure of [35], and the same proof techniques as in [35] can apply to show its security, generally show to be a SPHF following Definition 5[18]. The CPA-friendliness of this DCR-based Gennaro-Lindell SPHF is then a direct corollary of [18, Lemma 15].

Lemma 38 (Corollary of Lemma 15 of [18]). *The DCR-based Gennaro-Lindell SPHF from [43] satisfies the key-homomorphic property, and is FE-CPA-friendly.*

Finally, the $\mathsf{tPHF} = (\mathsf{hashkg}^{\mathsf{tb}}, \mathsf{projkg}^{\mathsf{tb}}, \mathsf{hash}^{\mathsf{tb}}, \mathsf{projhash}^{\mathsf{tb}})$ that meets the requirements of our transformation in Theorem 36 can be instantiated by [18, Lemma 3] based on DCR. Its proof for FE-CCA-friendliness can be found in [18, Lemma 29].

In summary, plugging the above components into our MIFE construction in Figure 5 we obtain a tag-based IND-CCA secure MIFE scheme for the function class $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$ under the DDH/DCR assumption. We obtain the following corollary from Theorem 36.

Corollary 39. *Under the* DDH*, or the* DCR*, assumption, there exists a* tag-based *IND-CCA secure MIFE scheme, following the security notion in Definition 48, for the function class* $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$.

Final steps. Assuming further the existence of one-time signatures and collision-resistant hash functions, we can achieve IND-CCA security, using similar technique as in [55]. The ideas are resumed prior to Theorem 36 of Section 5.1, in **Step 2b** of our main blueprint.

Remark 40. Considering the class $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$ to compute inner products, an attempt to go all the way from a single client IPFE to MIFE for inner products resembles the below:

$$\mathsf{IPFE} \overset{(1)}{\to} \text{secret-key } \mathsf{IPFE} \overset{(2)}{\to} \mathsf{MI\text{-}IPFE} \overset{(3)}{\to} \mathsf{MI\text{-}IPFE} \text{ w/ } \mathsf{Tag}$$

where step (1) privatizes the public key of MI-IPFE into some msk for encryption, assuming the keys can be decomposed for encrypting slots i independently[19] step (2) consists of decomposing the msk into multiple encryption keys ek_i, step (3) allows treating tags as simple as relying on the ROM during encryption (for deriving the encryption randomness, for example). Applying

[17] Equivalently N-residuosity is the 1-DCR and is implied by the general DCR, following Lemma 1

[18] The modifications of [43, Sect 8.5] is for better efficiency.

[19] This holds for many IPFE schemes, *e.g.* the famous DDH-based IPFE from [11] has this property, including the SPHF-based IPFE from [18].

this idea to the FE scheme from [18, 28] would lead to a tag-based IND-CCA MIFE for inner products[20]. In terms of ciphertxt's size, the induced tag-based MIFE from [18, 28] will contain $2n$ masked hash values (both the SPHF and tag-based SPHF) and n statements. On the other hand, our tag-based MIFE for $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$, when restricting to computing simple inner products, will contain $2n$ masked hash values and n statements. Our scheme matches the efficiency of the former [18, 28], *i.e.* total communication grows asymptotically linearly in the number of slots, allowing encrypting subvectors *as per* $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$, and we improve to fully adaptive security as the first IND-CCA-secure MIFE (the FE schemes from [18] only consider selective security regarding the challenge ciphertexts, which are later improved to be adaptively IND-CCA-secure in [28] while staying public key single client FE).

Acknowledgments

We thank David Pointcheval and Duong Hieu Phan for valuable remarks. This work was supported in part by the European Union Horizon 2020 ERC Programme (Grant Agreement no. 966570 – CryptAnalytics), the Beyond5G project and the French ANR Project ANR-19-CE39-0011 PRESTO.

References

1. Abdalla, M., Benhamouda, F., Blazy, O., Chevalier, C., Pointcheval, D.: SPHF-friendly non-interactive commitments. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 214–234. Springer, Heidelberg (Dec 2013). `https://doi.org/10.1007/978-3-642-42033-7_12`
2. Abdalla, M., Benhamouda, F., Gay, R.: From single-input to multi-client inner-product functional encryption. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 552–582. Springer, Heidelberg (Dec 2019). `https://doi.org/10.1007/978-3-030-34618-8_19`
3. Abdalla, M., Benhamouda, F., Kohlweiss, M., Waldner, H.: Decentralizing inner-product functional encryption. In: Lin, D., Sako, K. (eds.) PKC 2019, Part II. LNCS, vol. 11443, pp. 128–157. Springer, Heidelberg (Apr 2019). `https://doi.org/10.1007/978-3-030-17259-6_5`
4. Abdalla, M., Benhamouda, F., Pointcheval, D.: Disjunctions for hash proof systems: New constructions and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 69–100. Springer, Heidelberg (Apr 2015). `https://doi.org/10.1007/978-3-662-46803-6_3`
5. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 733–751. Springer, Heidelberg (Mar / Apr 2015). `https://doi.org/10.1007/978-3-662-46447-2_33`
6. Abdalla, M., Catalano, D., Fiore, D., Gay, R., Ursu, B.: Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 597–627. Springer, Heidelberg (Aug 2018). `https://doi.org/10.1007/978-3-319-96884-1_20`
7. Abdalla, M., Catalano, D., Gay, R., Ursu, B.: Inner-product functional encryption with fine-grained access control. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part III. LNCS, vol. 12493, pp. 467–497. Springer, Heidelberg (Dec 2020). `https://doi.org/10.1007/978-3-030-64840-4_16`
8. Agrawal, S., Goyal, R., Tomida, J.: Multi-input quadratic functional encryption from pairings. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 208–238. Springer, Heidelberg, Virtual Event (Aug 2021). `https://doi.org/10.1007/978-3-030-84259-8_8`
9. Agrawal, S., Goyal, R., Tomida, J.: Multi-party functional encryption. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part II. LNCS, vol. 13043, pp. 224–255. Springer, Heidelberg (Nov 2021). `https://doi.org/10.1007/978-3-030-90453-1_8`
10. Agrawal, S., Goyal, R., Tomida, J.: Multi-input quadratic functional encryption: Stronger security, broader functionality. In: Kiltz, E., Vaikuntanathan, V. (eds.) TCC 2022, Part I. LNCS, vol. 13747, pp. 711–740. Springer, Heidelberg (Nov 2022). `https://doi.org/10.1007/978-3-031-22318-1_25`
11. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 333–362. Springer, Heidelberg (Aug 2016). `https://doi.org/10.1007/978-3-662-53015-3_12`
12. Agrawal, S., Tomida, J., Yadav, A.: Attribute-based multi-input FE (and more) for attribute-weighted sums. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part IV. LNCS, vol. 14084, pp. 464–497. Springer, Heidelberg (Aug 2023). `https://doi.org/10.1007/978-3-031-38551-3_15`

[20] Their function class does not seem to allow encrypting subvectors, rather each complete encryption specifies a vector over $\mathbb{Z}_q^n$.

13. Agrawal, S., Tomida, J., Yadav, A.: Attribute-based multi-input fe (and more) for attribute-weighted sums (2023), https://eprint.iacr.org/2023/1191
14. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 308–326. Springer, Heidelberg (Aug 2015). https://doi.org/10.1007/978-3-662-47989-6_15
15. Ananth, P., Sahai, A.: Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 152–181. Springer, Heidelberg (Apr / May 2017). https://doi.org/10.1007/978-3-319-56620-7_6
16. Attrapadung, N., Libert, B., de Panafieu, E.: Expressive key-policy attribute-based encryption with constant-size ciphertexts. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 90–108. Springer, Heidelberg (Mar 2011). https://doi.org/10.1007/978-3-642-19379-8_6
17. Baltico, C.E.Z., Catalano, D., Fiore, D., Gay, R.: Practical functional encryption for quadratic functions with applications to predicate encryption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 67–98. Springer, Heidelberg (Aug 2017). https://doi.org/10.1007/978-3-319-63688-7_3
18. Benhamouda, F., Bourse, F., Lipmaa, H.: CCA-secure inner-product functional encryption from projective hash functions. In: Fehr, S. (ed.) PKC 2017, Part II. LNCS, vol. 10175, pp. 36–66. Springer, Heidelberg (Mar 2017). https://doi.org/10.1007/978-3-662-54388-7_2
19. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. In: Guruswami, V. (ed.) 56th FOCS. pp. 171–190. IEEE Computer Society Press (Oct 2015). https://doi.org/10.1109/FOCS.2015.20
20. Blazy, O., Fuchsbauer, G., Izabachène, M., Jambert, A., Sibert, H., Vergnaud, D.: Batch Groth-Sahai. In: Zhou, J., Yung, M. (eds.) ACNS 10. LNCS, vol. 6123, pp. 218–235. Springer, Heidelberg (Jun 2010). https://doi.org/10.1007/978-3-642-13708-2_14
21. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-44647-8_13
22. Boneh, D., Gentry, C., Hamburg, M.: Space-efficient identity based encryption without pairings. In: 48th FOCS. pp. 647–657. IEEE Computer Society Press (Oct 2007). https://doi.org/10.1109/FOCS.2007.64
23. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (Mar 2011). https://doi.org/10.1007/978-3-642-19571-6_16
24. Camenisch, J., Chandran, N., Shoup, V.: A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 351–368. Springer, Heidelberg (Apr 2009). https://doi.org/10.1007/978-3-642-01001-9_20
25. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-44647-8_2
26. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: 34th ACM STOC. pp. 494–503. ACM Press (May 2002). https://doi.org/10.1145/509907.509980
27. Castagnos, G., Laguillaumie, F., Tucker, I.: Practical fully secure unrestricted inner product functional encryption modulo p. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 733–764. Springer, Heidelberg (Dec 2018). https://doi.org/10.1007/978-3-030-03329-3_25
28. Castagnos, G., Laguillaumie, F., Tucker, I.: A tighter proof for CCA secure inner product functional encryption: Genericity meets efficiency. Theoretical Computer Science (2022). https://doi.org/10.1016/j.tcs.2022.02.014, https://inria.hal.science/hal-03780500
29. Chen, J., Lim, H.W., Ling, S., Wang, H., Wee, H.: Shorter IBE and signatures via asymmetric pairings. In: Abdalla, M., Lange, T. (eds.) PAIRING 2012. LNCS, vol. 7708, pp. 122–140. Springer, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-36334-4_8
30. Chotard, J., Dufour Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Decentralized multi-client functional encryption for inner product. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 703–732. Springer, Heidelberg (Dec 2018). https://doi.org/10.1007/978-3-030-03329-3_24
31. Chotard, J., Dufour Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Multi-client functional encryption with repetition for inner product. Cryptology ePrint Archive, Report 2018/1021 (2018), https://eprint.iacr.org/2018/1021
32. Chotard, J., Dufour-Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Dynamic decentralized functional encryption. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part I. LNCS, vol. 12170, pp. 747–775. Springer, Heidelberg (Aug 2020). https://doi.org/10.1007/978-3-030-56784-2_25
33. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) 8th IMA International Conference on Cryptography and Coding. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (Dec 2001)
34. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO'98. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (Aug 1998). https://doi.org/10.1007/BFb0055717

35. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (Apr / May 2002). https://doi.org/10.1007/3-540-46035-7_4
36. Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (Feb 2001). https://doi.org/10.1007/3-540-44586-2_9
37. Datta, P., Okamoto, T., Tomida, J.: Full-hiding (unbounded) multi-input inner product functional encryption from the k-Linear assumption. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 245–277. Springer, Heidelberg (Mar 2018). https://doi.org/10.1007/978-3-319-76581-5_9
38. Devevey, J., Libert, B., Peters, T.: Rational Modular Encoding in the DCR Setting: Non-Interactive Range Proofs and Paillier-Based Naor-Yung in the Standard Model. In: PKC 2022. Yokohama (virtual event), Japan (Mar 2022), https://hal.inria.fr/hal-03807457
39. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for Diffie-Hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 129–147. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40084-1_8
40. Faust, S., Kohlweiss, M., Marson, G.A., Venturi, D.: On the non-malleability of the Fiat-Shamir transform. In: Galbraith, S.D., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 60–79. Springer, Heidelberg (Dec 2012). https://doi.org/10.1007/978-3-642-34931-7_5
41. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (Aug 1987). https://doi.org/10.1007/3-540-47721-7_12
42. Gay, R.: A new paradigm for public-key functional encryption for degree-2 polynomials. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part I. LNCS, vol. 12110, pp. 95–120. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45374-9_4
43. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 524–543. Springer, Heidelberg (May 2003). https://doi.org/10.1007/3-540-39200-9_33, https://eprint.iacr.org/2003/032.ps.gz
44. Goldwasser, S., Gordon, S.D., Goyal, V., Jain, A., Katz, J., Liu, F.H., Sahai, A., Shi, E., Zhou, H.S.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 578–602. Springer, Heidelberg (May 2014). https://doi.org/10.1007/978-3-642-55220-5_32
45. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Predicate encryption for circuits from LWE. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 503–523. Springer, Heidelberg (Aug 2015). https://doi.org/10.1007/978-3-662-48000-7_25
46. Gordon, S.D., Katz, J., Liu, F.H., Shi, E., Zhou, H.S.: Multi-input functional encryption. Cryptology ePrint Archive, Report 2013/774 (2013), https://eprint.iacr.org/2013/774
47. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., De Capitani di Vimercati, S. (eds.) ACM CCS 2006. pp. 89–98. ACM Press (Oct / Nov 2006). https://doi.org/10.1145/1180405.1180418, available as Cryptology ePrint Archive Report 2006/309
48. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (Dec 2006). https://doi.org/10.1007/11935230_29
49. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (Apr 2008). https://doi.org/10.1007/978-3-540-78967-3_24
50. Hofheinz, D., Jager, T.: Tightly secure signatures and public-key encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 590–607. Springer, Heidelberg (Aug 2012). https://doi.org/10.1007/978-3-642-32009-5_35
51. Hofheinz, D., Jager, T.: Tightly secure signatures and public-key encryption. DCC **80**(1), 29–61 (2016). https://doi.org/10.1007/s10623-015-0062-x
52. Jutla, C.S., Roy, A.: Relatively-sound NIZKs and password-based key-exchange. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 485–503. Springer, Heidelberg (May 2012). https://doi.org/10.1007/978-3-642-30057-8_29
53. Katz, J., Vaikuntanathan, V.: Round-optimal password-based authenticated key exchange. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 293–310. Springer, Heidelberg (Mar 2011). https://doi.org/10.1007/978-3-642-19571-6_18
54. Katz, J., Vaikuntanathan, V.: Round-optimal password-based authenticated key exchange. Journal of Cryptology **26**(4), 714–743 (Oct 2013). https://doi.org/10.1007/s00145-012-9133-6
55. Kiltz, E.: Chosen-ciphertext security from tag-based encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 581–600. Springer, Heidelberg (Mar 2006). https://doi.org/10.1007/11681878_30
56. Kiltz, E., Vahlis, Y.: CCA2 secure IBE: Standard model efficiency through authenticated symmetric encryption. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 221–238. Springer, Heidelberg (Apr 2008). https://doi.org/10.1007/978-3-540-79263-5_14

57. Lewko, A.B., Waters, B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (Feb 2010). https://doi.org/10.1007/978-3-642-11799-2_27
58. Libert, B., Peters, T., Joye, M., Yung, M.: Non-malleability from malleability: Simulation-sound quasi-adaptive NIZK proofs and CCA2-secure encryption from homomorphic signatures. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 514–532. Springer, Heidelberg (May 2014). https://doi.org/10.1007/978-3-642-55220-5_29
59. Libert, B., Titiu, R.: Multi-client functional encryption for linear functions in the standard model from LWE. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 520–551. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-34618-8_18
60. Libert, B., Yung, M.: Non-interactive CCA-secure threshold cryptosystems with adaptive security: New framework and constructions. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 75–93. Springer, Heidelberg (Mar 2012). https://doi.org/10.1007/978-3-642-28914-9_5
61. Lin, H.: Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 599–629. Springer, Heidelberg (Aug 2017). https://doi.org/10.1007/978-3-319-63688-7_20
62. Nandi, M., Pandit, T.: Generic conversions from CPA to CCA secure functional encryption. Cryptology ePrint Archive, Report 2015/457 (2015), https://eprint.iacr.org/2015/457
63. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: 22nd ACM STOC. pp. 427–437. ACM Press (May 1990). https://doi.org/10.1145/100216.100273
64. Nguyen, D.: Dynamic decentralized functional encryptions from pairings in the standard model. Cryptology ePrint Archive, Paper 2024/580 (2024), https://eprint.iacr.org/2024/580
65. Nguyen, K., Phan, D.H., Pointcheval, D.: Multi-client functional encryption with fine-grained access control. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part I. LNCS, vol. 13791, pp. 95–125. Springer, Heidelberg (Dec 2022). https://doi.org/10.1007/978-3-031-22963-3_4
66. Nguyen, K., Phan, D.H., Pointcheval, D.: Optimal security notion for decentralized multi-client functional encryption. In: Tibouchi, M., Wang, X. (eds.) ACNS 23, Part II. LNCS, vol. 13906, pp. 336–365. Springer, Heidelberg (Jun 2023). https://doi.org/10.1007/978-3-031-33491-7_13
67. Nguyen, K., Phan, D.H., Pointcheval, D.: Multi-client functional encryption with public inputs and strong security. Cryptology ePrint Archive, Paper 2024/740 (2024), https://eprint.iacr.org/2024/740
68. Nguyen, K., Pointcheval, D., Schädlich, R.: Dynamic decentralized functional encryption:generic constructions with strong security (2024)
69. Nguyen, K., Pointcheval, D., Schädlich, R.: Decentralized multi-client functional encryption with strong security. IACR Communications in Cryptology **1**(2) (2024). https://doi.org/10.62056/andkp2fgx
70. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (Aug 2010). https://doi.org/10.1007/978-3-642-14623-7_11
71. Okamoto, T., Takashima, K.: Adaptively attribute-hiding (hierarchical) inner product encryption. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 591–608. Springer, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_35
72. Okamoto, T., Takashima, K.: Fully secure unbounded inner-product and attribute-based encryption. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 349–366. Springer, Heidelberg (Dec 2012). https://doi.org/10.1007/978-3-642-34961-4_22
73. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: Ning, P., De Capitani di Vimercati, S., Syverson, P.F. (eds.) ACM CCS 2007. pp. 195–203. ACM Press (Oct 2007). https://doi.org/10.1145/1315245.1315270
74. de Perthuis, P., Pointcheval, D.: Two-client inner-product functional encryption with an application to money-laundering detection. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 725–737. ACM Press (Nov 2022). https://doi.org/10.1145/3548606.3559374
75. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th FOCS. pp. 543–553. IEEE Computer Society Press (Oct 1999). https://doi.org/10.1109/SFFCS.1999.814628
76. Sahai, A., Waters, B.R.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (May 2005). https://doi.org/10.1007/11426639_27
77. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO'84. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (Aug 1984)
78. Shi, E., Vanjani, N.: Multi-client inner product encryption: Function-hiding instantiations without random oracles. In: Boldyreva, A., Kolesnikov, V. (eds.) PKC 2023, Part I. LNCS, vol. 13940, pp. 622–651. Springer, Heidelberg (May 2023). https://doi.org/10.1007/978-3-031-31368-4_22
79. Shoup, V.: Why chosen ciphertext security matters. IBM Research Report (1998), https://dominoweb.draco.res.ibm.com/7e2d4b9b2c2644bc852566bb003a3b67.html
80. Tompa, M., Woll, H.: Random self-reducibility and zero knowledge interactive proofs of possession of information. In: 28th FOCS. pp. 472–482. IEEE Computer Society Press (Oct 1987). https://doi.org/10.1109/SFCS.1987.49

81. Waters, B.: Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (Aug 2009). https://doi.org/10.1007/978-3-642-03356-8_36
82. Yamada, S., Attrapadung, N., Hanaoka, G., Kunihiro, N.: Generic constructions for chosen-ciphertext secure attribute based encryption. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 71–89. Springer, Heidelberg (Mar 2011). https://doi.org/10.1007/978-3-642-19379-8_5

A Additional Definitions

A.1 Hardness Assumptions

We state the assumptions needed for our constructions.

Definition 41. *In a cyclic group $\mathbb{G}$ of prime order q, the* **Decisional Diffie-Hellman** (DDH) *problem is to distinguish the distributions*

$$D_0 = \{(\llbracket 1 \rrbracket, \llbracket a \rrbracket, \llbracket b \rrbracket, \llbracket ab \rrbracket)\} \qquad\qquad D_1 = \{(\llbracket 1 \rrbracket, \llbracket a \rrbracket, \llbracket b \rrbracket, \llbracket c \rrbracket)\}.$$

for $a, b, c \xleftarrow{\$} \mathbb{Z}_q$. The DDH *assumption in $\mathbb{G}$ assumes that no* ppt *adversary can solve the* DDH *problem with non-negligible probability.*

Definition 42. *In the bilinear setting $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, \mathbf{e}, q)$, the* **Symmetric eXternal Diffie-Hellman** (SXDH) *assumption makes the* DDH *assumption in both $\mathbb{G}_1$ and $\mathbb{G}_2$.*

A.2 Dual Pairing Vector Spaces

Our constructions rely on the *Dual Pairing Vector Spaces* (DPVS) framework in prime-order bilinear group setting $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, \mathbf{e}, q)$ and $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$ are all written additively. The DPVS technique dates back to the seminal work by Okamoto-Takashima [70, 71, 72] aiming at adaptive security for ABE as well as IBE, together with the *dual system methodology* introduced by Waters [81]. In [57], the setting for dual systems is composite-order bilinear groups. Continuing on this line of works, Chen *et al.* [29] used prime-order bilinear groups under the SXDH assumption. Let us fix $N \in \mathbb{N}$ and consider $\mathbb{G}_1^N$ having N copies of $\mathbb{G}_1$. Any $\mathbf{x} = \llbracket (x_1, \dots, x_N) \rrbracket_1 \in \mathbb{G}_1^N$ is identified as the vector $(x_1, \dots, x_N) \in \mathbb{Z}_q^N$. There is no ambiguity because $\mathbb{G}_1$ is a cyclic group of order q prime. The $\mathbf{0}$-vector is $\mathbf{0} = \llbracket (0, \dots, 0) \rrbracket_1$. The addition of two vectors in $\mathbb{G}_1^N$ is defined by coordinate-wise addition. The scalar multiplication of a vector is defined by $t \cdot \mathbf{x} := \llbracket t \cdot (x_1, \dots, x_N) \rrbracket_1$, where $t \in \mathbb{Z}_q$ and $\mathbf{x} = \llbracket (x_1, \dots, x_N) \rrbracket_1$. The additive inverse of $\mathbf{x} \in \mathbb{G}_1^N$ is defined to be $-\mathbf{x} := \llbracket (-x_1, \dots, -x_N) \rrbracket_1$. Viewing $\mathbb{Z}_q^N$ as a vector space of dimension N over $\mathbb{Z}_q$ with the notions of bases, we can obtain naturally a similar notion of bases for $\mathbb{G}_1^N$. More specifically, any invertible matrix $B \in GL_N(\mathbb{Z}_q)$ identifies a basis $\mathbf{B}$ of $\mathbb{G}_1^N$, whose i-th row $\mathbf{b}_i$ is $\llbracket B^{(i)} \rrbracket_1$, where $B^{(i)}$ is the i-th row of B. The canonical basis $\mathbf{A}$ of $\mathbb{G}_1^N$ consists of $\mathbf{a}_1 := \llbracket (1, 0 \dots, 0) \rrbracket_1, \mathbf{a}_2 := \llbracket (0, 1, 0 \dots, 0) \rrbracket_1, \dots, \mathbf{a}_N := \llbracket (0, \dots, 0, 1) \rrbracket_1$. It is straightforward that we can write $\mathbf{B} = B \cdot \mathbf{A}$ for any basis $\mathbf{B}$ of $\mathbb{G}_1^N$ corresponding to an invertible matrix $B \in GL_N(\mathbb{Z}_q)$. We write $\mathbf{x} = (x_1, \dots, x_N)_{\mathbf{B}}$ to indicate the representation of $\mathbf{x}$ in the basis $\mathbf{B}$, i.e. $\mathbf{x} = \sum_{i=1}^N x_i \cdot \mathbf{b}_i$. By convention the writing $\mathbf{x} = (x_1, \dots, x_N)$ concerns the canonical basis $\mathbf{A}$.

Treating $\mathbb{G}_2^N$ similarly, we can furthermore define a product of two vectors $\mathbf{x} = \llbracket (x_1, \dots, x_N) \rrbracket_1 \in \mathbb{G}_1^N, \mathbf{y} = \llbracket (y_1, \dots, y_N) \rrbracket_2 \in \mathbb{G}_2^N$ by $\mathbf{x} \times \mathbf{y} := \prod_{i=1}^N \mathbf{e}(\mathbf{x}[i], \mathbf{y}[i]) = \llbracket \langle (x_1, \dots, x_N), (y_1, \dots, y_N) \rangle \rrbracket_t$. Given a basis $\mathbf{B} = (\mathbf{b}_i)_{i \in [N]}$ of $\mathbb{G}_1^N$, we define $\mathbf{B}^*$ to be a basis of $\mathbb{G}_2^N$ by first defining $B' := (B^{-1})^\top$ and the i-th row $\mathbf{b}_i^*$ of $\mathbf{B}^*$ is $\llbracket B'^{(i)} \rrbracket_2$. It holds that $B \cdot (B')^\top = I_N$ the identity matrix and $\mathbf{b}_i \times \mathbf{b}_j^* = \llbracket \delta_{i,j} \rrbracket_t$ for every $i, j \in [N]$, where $\delta_{i,j} = 1$ if and only if $i = j$. We call the pair $(\mathbf{B}, \mathbf{B}^*)$ a *pair of dual orthogonal bases* of $(\mathbb{G}_1^N, \mathbb{G}_2^N)$. If $\mathbf{B}$ is constructed by a random invertible matrix $B \xleftarrow{\$} GL_N(\mathbb{Z}_q)$, we call the resulting $(\mathbf{B}, \mathbf{B}^*)$ a pair of random dual bases. A DPVS is a bilinear group setting $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, \mathbf{e}, q, N)$ with dual orthogonal bases. In this work, we also use extensively *basis changes* over dual orthogonal bases of a DPVS to argue the steps of switching key as well as ciphertext vectors to semi-functional mode in our proofs. The details of such basis changes are recalled in the appendix A.3.

A.3 Dual Pairing Vector Spaces

Basis Changes. In this work, we use extensively *basis changes* over dual orthogonal bases of a DPVS. We again use $\mathbb{G}_1^N$ as a running example. Let $(\mathbf{A}, \mathbf{A}^*)$ be the dual canonical bases of $(\mathbb{G}_1^N, \mathbb{G}_2^N)$. Let $(\mathbf{U} = (\mathbf{u}_i)_i, \mathbf{U}^* = (\mathbf{u}_i^*)_i)$ be a pair of dual bases of $(\mathbb{G}_1^N, \mathbb{G}_2^N)$, corresponding to an invertible matrix $U \in \mathbb{Z}_q^{N\times N}$. Given an invertible matrix $B \in \mathbb{Z}_q^{N\times N}$, the basis change from $\mathbf{U}$ w.r.t B is defined to be $\mathbf{B} \coloneqq B \cdot \mathbf{U}$, which means:

$$\begin{aligned}(x_1, \ldots, x_N)_{\mathbf{B}} &= \sum_{i=1}^{N} x_i \mathbf{b}_i = (x_1, \ldots, x_N) \cdot \mathbf{B} = (x_1, \ldots, x_N) \cdot B \cdot \mathbf{U} \\ &= (y_1, \ldots, y_N)_{\mathbf{U}} \text{ where } (y_1, \ldots, y_N) \coloneqq (x_1, \ldots, x_N) \cdot B \ .\end{aligned}$$

Under a basis change $\mathbf{B} = B \cdot \mathbf{U}$, we have

$$(x_1, \ldots, x_N)_{\mathbf{B}} = ((x_1, \ldots, x_N) \cdot B)_{\mathbf{U}} \ ; \ (y_1, \ldots, y_N)_{\mathbf{U}} = \Big((y_1, \ldots, y_N) \cdot B^{-1}\Big)_{\mathbf{B}} \quad . \tag{11}$$

The computation is extended to the dual basis change $\mathbf{B}^* = B' \cdot \mathbf{U}^*$, where $B' = \left(B^{-1}\right)^\top$:

$$(x_1, \ldots, x_N)_{\mathbf{B}^*} = \Big((x_1, \ldots, x_N) \cdot B'\Big)_{\mathbf{U}^*} \ ; \ (y_1, \ldots, y_N)_{\mathbf{U}^*} = \Big((y_1, \ldots, y_N) \cdot B^\top\Big)_{\mathbf{B}^*} \quad . \tag{12}$$

It can be checked that $(\mathbf{B}, \mathbf{B}^*)$ remains a pair of dual orthogonal bases. When we consider a basis change $\mathbf{B} = B \cdot \mathbf{U}$, if $B = (b_{i,j})_{i,j}$ affects only a subset $J \subseteq [N]$ of indices in the representation w.r.t basis $\mathbf{U}$, we will write B as the square block containing $(b_{i,j})_{i,j}$ for $i, j \in J$ and implicitly the entries of B outside this block are taken from the identity matrix I_N.

The basis changes are particularly useful in our security proofs. Intuitively these changes constitute a transition from a hybrid G having vectors expressed in $(\mathbf{U}, \mathbf{U}^*)$ to the next hybrid $\mathsf{G}_{\mathsf{next}}$ having vectors expressed in $(\mathbf{B}, \mathbf{B}^*)$. We focus on two types of basis changes, which are elaborated below. For simplicity, we consider dimension $N = 2$:

Formal Basis Change: We change $(\mathbf{U}, \mathbf{U}^*)$ into $(\mathbf{B}, \mathbf{B}^*)$ using

$$B \coloneqq \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}_{1,2} \qquad B' \coloneqq \left(B^{-1}\right)^\top = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}_{1,2}$$

$$\mathbf{B} = B \cdot \mathbf{U} \qquad \mathbf{B}^* = B' \cdot \mathbf{U}^* \ .$$

We use this type in situations such as: in G we have vectors *all* of the form $(x_1, 0)_{\mathbf{U}}, (y_1, 0)_{\mathbf{U}^*}$, and we want to go to $\mathsf{G}_{\mathsf{next}}$ having vectors *all* of the form $(x_1, 0)_{\mathbf{B}}, (y_1, \boxed{y_1})_{\mathbf{B}^*}$. The simulator writes *all* vectors $(x_1, 0)_{\mathbf{U}}, (y_1, 0)_{\mathbf{U}^*}$ in $(\mathbf{U}, \mathbf{U}^*)$ and under this basis change they are written into

$$(x_1, \ 0)_{\mathbf{U}} = (x_1 - 0, \ 0)_{\mathbf{B}} = (x_1, \ 0)_{\mathbf{B}}; \quad (y_1, \ 0)_{\mathbf{U}^*} = (y_1, \ 0 + y_1)_{\mathbf{B}^*} = (y_1, \ y_1)_{\mathbf{B}^*}$$

following the calculations in (11) and (12). The products between two dual vectors are invariant, *all* vectors are formally written from $(\mathbf{U}, \mathbf{U}^*)$ (corresponding to G) to $(\mathbf{B}, \mathbf{B}^*)$ (corresponding to $\mathsf{G}_{\mathsf{next}}$), the adversary's view over the vectors is thus identical from G to $\mathsf{G}_{\mathsf{next}}$. In particular, this is a kind of *information-theoretic property* of DPVS by basis changing that we exploit to have identical hybrids' hop in the security proof.

Computational Basis Change: Given an instance of a computational problem, *e.g.* $[\![(a, b, c)]\!]_1$ of DDH in $\mathbb{G}_1$ where $c - ab = 0$ or $\delta \xleftarrow{\$} \mathbb{Z}_q$, we change $(\mathbf{U}, \mathbf{U}^*)$ into $(\mathbf{B}, \mathbf{B}^*)$ using

$$B \coloneqq \begin{bmatrix} 1 & 0 \\ a & 1 \end{bmatrix}_{1,2} \qquad B' \coloneqq \left(B^{-1}\right)^\top = \begin{bmatrix} 1 & -a \\ 0 & 1 \end{bmatrix}_{1,2}$$

$$\mathbf{B} = B \cdot \mathbf{U} \qquad \mathbf{B}^* = B' \cdot \mathbf{U}^* \ .$$

One situation where this type of basis change can be useful is: in G we have *some* target vectors of the form $(0, \mathsf{rnd})_{\mathbf{U}}$, where $\mathsf{rnd} \xleftarrow{\$} \mathbb{Z}_q$ is a random scalar, together with other $(z_1, z_2)_{\mathbf{U}}$, and *all* the dual is of the form $(0,\ y_2)_{\mathbf{U}^*}$. We want to go to $\mathsf{G}_{\mathsf{next}}$ having $(\boxed{\widetilde{\mathsf{rnd}}},\ \mathsf{rnd})_{\mathbf{B}}$ masked by some randomness $\widetilde{\mathsf{rnd}} \xleftarrow{\$} \mathbb{Z}_q$, while keeping $(0,\ y_2)_{\mathbf{B}^*}$. Because $[\![a]\!]_1$ is given, the simulator can simulate vectors $(z_1, z_2)_{\mathbf{U}}$ directly in $\mathbf{B}$ using $[\![a]\!]_1$ as well as the known coordinates z_1, z_2. The basis change will be employed for the simulation of target vectors:

$$(c,\ b)_{\mathbf{U}} + (0, \mathsf{rnd})_{\mathbf{B}} = (c - a \cdot b,\ \ \mathsf{rnd} + b)_{\mathbf{B}};$$
$$(0,\ y_2)_{\mathbf{U}^*} = (0,\ y_2 + a \cdot 0)_{\mathbf{B}^*} = (0,\ y_2)_{\mathbf{B}^*}$$

where *all* vectors in $\mathbf{B}^*$ must be written first in $\mathbf{U}^*$, since we do not have $[\![a]\!]_2$, to see how the basis change affects them. Using the basis change we simulate those target vectors by $(c - a \cdot b,\ \mathsf{rnd} + b)_{\mathbf{B}}$ with rnd implicitly being updated to $\mathsf{rnd} + b$, the uninterested $(z_1, z_2)_{\mathbf{B}}$ are simulated correctly in $\mathbf{B}$, meanwhile the dual vectors $(0,\ y_2)_{\mathbf{B}^*}$ stays the same. Depending on the DDH instance, if $c - ab = 0$ the target vectors are in fact $(0,\ \mathsf{rnd})_{\mathbf{B}}$ and we are simulating G, else $c - ab = \delta \xleftarrow{\$} \mathbb{Z}_q$ the target vectors are simulated for $\mathsf{G}_{\mathsf{next}}$ and $\widetilde{\mathsf{rnd}} := \delta$. Hence, under the hardness of DDH in $\mathbb{G}_1$, a computationally bounded adversary cannot distinguish its views in the hybrids' hop from G to $\mathsf{G}_{\mathsf{next}}$.

We remark that the basis changes will modify basis vectors and for the indistinguishability to hold, perfectly in *formal* change and computationally in *computational* changes, all impacted basis vectors must not be revealed to the adversary.

Additional Notations. Any $\mathbf{x} = [\![(m_1, \ldots, m_N)]\!]_1 \in \mathbb{G}_1^N$ is identified as the vector $(m_1, \ldots, m_N) \in \mathbb{Z}_q^N$. There is no ambiguity because $\mathbb{G}_1$ is a cyclic group of order q prime. The $\mathbf{0}$-vector is $\mathbf{0} = [\![(0, \ldots, 0)]\!]_1$. The addition of two vectors in $\mathbb{G}_1^N$ is defined by coordinate-wise addition. The scalar multiplication of a vector is defined by $t \cdot \mathbf{x} := [\![t \cdot (m_1, \ldots, m_N)]\!]_1$, where $t \in \mathbb{Z}_q$ and $\mathbf{x} = [\![(m_1, \ldots, m_N)]\!]_1$. The additive inverse of $\mathbf{x} \in \mathbb{G}_1^N$ is defined to be $-\mathbf{x} := [\![(-m_1, \ldots, -m_N)]\!]_1$. The canonical basis $\mathbf{A}$ of $\mathbb{G}_1^N$ consists of $\mathbf{a}_1 := [\![(1, 0 \ldots, 0)]\!]_1, \mathbf{a}_2 := [\![(0, 1, 0 \ldots, 0)]\!]_1, \ldots, \mathbf{a}_N := [\![(0, \ldots, 0, 1)]\!]_1$. By convention the writing $\mathbf{x} = (m_1, \ldots, m_N)$ concerns the canonical basis $\mathbf{A}$.

A.4 More Cryptographic Primitives

We recall necessary cryptographic primitives used in this work.

Functional Encryption (FE).

Definition 43. *A* functional encryption *scheme for a class $\mathcal{F}$ is defined by a tuple of algorithms* $(\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$. *The* Setup *algorithm takes as input a security parameter 1^λ and outputs a public key* pk *and a master secret key* msk. *The* $\mathsf{Extract}$ *algorithm takes as input a master secret key* msk *and a function description $F_\lambda : \mathcal{M}_\lambda \to \mathcal{R}_\lambda$, and outputs a secret key* sk_F. *The* Enc *algorithm takes as input a public key* pk, *a message m in some message space $\mathcal{M}$, outputs a ciphertext* ct. *The* Dec *algorithm takes as input a secret key* sk_F *and a ciphertext* ct, *and outputs an element in $\mathcal{R}$. An FE for a class $\mathcal{F}$ is* correct *if for all $\lambda \in \mathbb{N}$, all* $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, *all $F_\lambda \in \mathcal{F}$, all $m \in \mathcal{M}$, and all* $\mathsf{sk}_F \leftarrow \mathsf{Keygen}(F_\lambda, \mathsf{msk})$, *it holds that* $\mathsf{Dec}(\mathsf{sk}_F, \mathsf{Enc}(\mathsf{pk}, m)) = F_\lambda(m)$.

The *security* of an FE scheme is defined below.

Definition 44. *A FE scheme $\mathcal{E}$ with respect to a class of functions $\mathcal{F}$ is* CPA-secure *if for every* ppt *adversary $\mathcal{A}$, the following probability is negligible in λ:*

$$\mathsf{Adv}^{\mathsf{fe}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) := \left| \Pr[\mathsf{Expr}^{\mathsf{fe}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) = 1] - \frac{1}{2} \right|$$

where the experiment $\mathsf{Expr}^{\mathsf{fe}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ *is defined as follows:*

1. *The challenger runs* $\mathsf{Setup}(1^\lambda)$ *to obtain* $(\mathsf{pk}, \mathsf{msk})$ *and outputs* pk *to* $\mathcal{A}$. *In the following the adversary* $\mathcal{A}$ *can make queries adaptively in any order before Finalize.*
2. *(Key queries) The adversary* $\mathcal{A}$ *adaptively outputs a function description* F_λ. *The challenger runs* $\mathsf{sk}_F \leftarrow \mathsf{Extract}(F_\lambda, \mathsf{msk})$ *and returns* sk_F *to* $\mathcal{A}$.
3. *(Decryption) The adversary* $\mathcal{A}$ *adaptively outputs* (ct, F_λ) *containing a ciphertext* ct *and a function description* F_λ. *The challenger generates the functional key* $\mathsf{sk}_F \leftarrow \mathsf{Extract}(F_\lambda, \mathsf{msk})$ *and returns* $\mathsf{Dec}(\mathsf{sk}_F, \mathsf{ct})$ *to* $\mathcal{A}$.
4. *(Challenge) The adversary* $\mathcal{A}$ *outputs a pair of messages* (m_0, m_1). *The challenger chooses a bit* $b \in \{0,1\}$ *and runs* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$.
5. *(Finalize) The adversary* $\mathcal{A}$ *outputs a guess* $\hat{b}$. *If there exists a function description* F_λ *such that* $F(m_0) \neq F(m_1)$, *then the expriment outputs* $\frac{1}{2}$. *Otherwise, the experiment outputs* $\hat{b} \overset{?}{=} b$.

We can define similar weaker notions of *selective* challenge message and/or *selective* functional decryptionkey queries. The notion of FE with *access control* can be captured by considering the class $\mathcal{F}$ that does not only include the calulating function F_λ, but also the access control policies $\mathbb{A}$ given any member $(F_\lambda, \mathbb{A})$ in $\mathcal{F}$. The *correctness* is adapted that the decryption key $\mathsf{sk}_{F,\mathbb{A}}$ can only decrypt the ciphertexts ct to $F(m)$ if the access control policy $\mathbb{A}$ accepts the attributes S of the ciphertext $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m, \mathsf{S})$. The notion of *security* is defined similarly as Definition 44, except that the syntax is adapted to the FE with access control.

Multi-Input Functional Encryption (MIFE). The notion of *Multi-Inputs Functional Encryption* (MIFE) is introduced in [44, 46], where the multi-ary function evaluates on a list of inputs. In this setting, a *single* user encrypts the various inputs at different slots. Evaluation of the function performed on the joint-inputs, using a functional decryption key generated by a trusted authority

Definition 45. *A* multi-input functional encryption *scheme is defined by a tuple of algorithms* $(\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$. *The* Setup *algorithm takes as input a security parameter* 1^λ *and a number of slots* n, *and outputs a public parameter* pp, *a master secret key* msk, *and* n *encryption keys* ek_i. *The* $\mathsf{Extract}$ *algorithm takes as input a function description* $F_\lambda : \prod_{i=1}^{n} \mathcal{D}_{\lambda,i} \to \mathcal{R}_\lambda$ *and the master secret key* msk, *and outputs a decryption key* dk_F. *The* Enc *algorithm takes as input an encryption key* ek_i *and a message* m_i *in some message space* $\mathcal{D}_{\lambda,i}$, *and outputs a ciphertext* ct_i. *The* Dec *algorithm takes as input a decryption key* dk_F *and a vector of ciphertexts* ct_i *of length* n, *and outputs an element in* $\mathcal{R}_\lambda$ *or* $\perp$. *An MIFE for a class* $\mathcal{F}$ *is* correct *if for all* $\lambda \in \mathbb{N}$, *all* $(\mathsf{pp}, \mathsf{msk}, (\mathsf{ek}_i)_{i\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, *all* $F_\lambda \in \mathcal{F}$, *all* $m_i \in \mathcal{D}_{\lambda,i}$, *and all* $\mathsf{dk}_F \leftarrow \mathsf{Extract}(F_\lambda, \mathsf{msk})$, *it holds that* $\mathsf{Dec}(\mathsf{dk}_{F_\lambda}, (\mathsf{Enc}(\mathsf{ek}_i, m_i))_{i\in[n]}) = F_\lambda(m_i)_{i\in[n]}$.

The *security* of an MIFE is defined below.

Definition 46. *An MIFE scheme* $\mathcal{E}$ *with respect to a class of functions* $\mathcal{F}$ *is* secure *if for every* ppt *adversary* $\mathcal{A}$, *the following probability is negligible in* λ:

$$\mathsf{Adv}^{\mathsf{mife}}_{\mathcal{F},\mathcal{A}}(1^\lambda) := \left| \Pr[\mathsf{Expr}^{\mathsf{mife}}_{\mathcal{F},\mathcal{A}}(1^\lambda) = 1] - \frac{1}{2} \right|$$

where the experiment $\mathsf{Expr}^{\mathsf{mife}}_{\mathcal{F},\mathcal{A}}(1^\lambda)$ *is defined as follows:*

1. *The challenger runs* $\mathsf{Setup}(1^\lambda, 1^n)$ *to obtain* $(\mathsf{pp}, \mathsf{msk}, (\mathsf{ek}_i)_{i\in[n]})$ *and outputs* pp *to* $\mathcal{A}$. *In the following the adversary* $\mathcal{A}$ *can make queries adaptively in any order before Finalize.*
2. *(Key queries) The adversary* $\mathcal{A}$ *adaptively outputs a function description* F_λ. *The challenger runs* $\mathsf{dk}_F \leftarrow \mathsf{Extract}(F_\lambda, \mathsf{msk})$ *and returns* dk_F *to* $\mathcal{A}$.
3. *(Challenge) The adversary* $\mathcal{A}$ *outputs a query* $(i, m_i^{(0)}, m_i^{(1)})$ *for some* $i \in [n]$. *The challenger chooses a bit* $b \in \{0,1\}$ *and encrypts* $m_i^{(b)}$ *to obtain* $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, m_i^{(b)})$. *The ciphertext* ct_i *is returned to* $\mathcal{A}$.

4. *(Encryption) The adversary $\mathcal{A}$ outputs a query (i, m_i) for some $i \in [n]$. The challenger encrypts m_i to obtain $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, m_i)$. The ciphertext ct_i is returned to $\mathcal{A}$.*
5. *(Finalize) The adversary $\mathcal{A}$ outputs a guess $\hat{b}$. If the following conditions is satisfied, the experiment outputs $\hat{b} \overset{?}{=} b$. Otherwise, the experiment outputs 0. Let $I \subset [n]$ be the set of corrupted indices, for $b \in \{0,1\}$ we define $\mathbf{X}^{(b)} := \{x_{1,j}^{(b)}, \ldots, x_{n,j}^{(b)}\}_{j=1}^{q}$ to be the q queried challenges*
   (a) *The pair $\mathbf{X}^{(0)}, \mathbf{X}^{(1)}$ satisfies that for all F queried by $\mathcal{A}$, all $I' = \{i_1, \ldots, i_t\} \subseteq I$, all $\{x'_{i_1}, \ldots, x'_{i_t}\}$, all $j_1, \ldots, j_{n-t} \in [q]$ we have*
   $$F\left(\mathsf{order}\left(x_{i_1,j_1}^{(0)}, \ldots, x_{i_{n-t},j_{n-t}}^{(0)}, x'_{i_1}, \ldots, x'_{i_t}\right)\right) \\ = F\left(\mathsf{order}\left(x_{i_1,j_1}^{(1)}, \ldots, x_{i_{n-t},j_{n-t}}^{(1)}, x'_{i_1}, \ldots, x'_{i_t}\right)\right)$$
   (b) *The set $\{F\}$ queried by $\mathcal{A}$ satisfies that for all $\mathbf{X}^{(0)}, \mathbf{X}^{(1)}$ challenges, all $I' = \{i_1, \ldots, i_t\} \subseteq I$, all $\{x'_{i_1}, \ldots, x'_{i_t}\}$, all $j_1, \ldots, j_{n-t} \in [q]$ we have*
   $$F\left(\mathsf{order}\left(x_{i_1,j_1}^{(0)}, \ldots, x_{i_{n-t},j_{n-t}}^{(0)}, x'_{i_1}, \ldots, x'_{i_t}\right)\right) \\ = F\left(\mathsf{order}\left(x_{i_1,j_1}^{(1)}, \ldots, x_{i_{n-t},j_{n-t}}^{(1)}, x'_{i_1}, \ldots, x'_{i_t}\right)\right)$$

   *such that the ℓ-input receives its correspond value by the permutation $\mathsf{order}(\cdot)$. Intuitively the set of inputs $\{x'_{i_1}, \ldots, x'_{i_t}\}$ represents whatever the adversary can put into the (subsets of) corrupted slots, and syntactically we use the permutation $\mathsf{order}(\cdot)$ to map values to their correct ordered arguments of the function (*e.g. *input value x'_{i_1} to argument k if $i_1 = k \in \mathbb{N}$).*

We can define similar weaker notions of *selective* challenge message and/or *selective* functional decryption key queries. The notion of MIFE with *access control* can be done in the same manner as we do for FE with access control in the previous paragraph. The *correctness* is adapted that the decryption key $\mathsf{sk}_{F,\mathbb{A}}$ can only decrypt the ciphertexts $(\mathsf{ct}_i)_i$ to $F((m_i)_i)$ if the access control policy $\mathbb{A}$ accepts the attributes S_i of the ciphertext $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, m_i, \mathsf{S}_i)$ for all slots $i \in [n]$.

Finally, we can also derive a CCA-security notion for MIFE following the introduced notion for MCFE in Definition 30. The following CCA-security notion is used throughout Section 5 for our IND-CCA secure MIFE construction. The *security* of an MIFE is defined below.

Definition 47 (CCA security). *An MIFE scheme $\mathcal{E}$ with respect to a class of functions $\mathcal{F}$ is* secure *if for every* ppt *adversary $\mathcal{A}$, the following probability is negligible in λ:*

$$\mathsf{Adv}_{\mathcal{F},\mathcal{A}}^{\mathsf{mife}}(1^\lambda) := \left|\Pr[\mathsf{Expr}_{\mathcal{F},\mathcal{A}}^{\mathsf{mife}}(1^\lambda) = 1] - \frac{1}{2}\right|$$

where the experiment $\mathsf{Expr}_{\mathcal{F},\mathcal{A}}^{\mathsf{mife}}(1^\lambda)$ is defined as follows:

1. *The challenger runs $\mathsf{Setup}(1^\lambda, 1^n)$ to obtain $(\mathsf{pp}, \mathsf{msk}, (\mathsf{ek}_i)_{i\in[n]})$ and outputs pp to $\mathcal{A}$. In the following the adversary $\mathcal{A}$ can make queries adaptively in any order before Finalize.*
2. *(Key queries) The adversary $\mathcal{A}$ adaptively outputs a function description F_λ. The challenger runs $\mathsf{dk}_F \leftarrow \mathsf{Extract}(F_\lambda, \mathsf{msk})$ and returns dk_F to $\mathcal{A}$.*
3. *(Challenge) The adversary $\mathcal{A}$ outputs a query $(i, m_i^{(0)}, m_i^{(1)})$ for some $i \in [n]$. The challenger chooses a bit $b \in \{0,1\}$ and encrypts $m_i^{(b)}$ to obtain $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, m_i^{(b)})$. The ciphertext ct_i is returned to $\mathcal{A}$.*
4. *(Encryption) The adversary $\mathcal{A}$ outputs a query (i, m_i) for some $i \in [n]$. The challenger encrypts m_i to obtain $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, m_i)$. The ciphertext ct_i is returned to $\mathcal{A}$.*
5. *(Decryption) The adversary $\mathcal{A}$ outputs at set of ciphertext $(\mathsf{ct}_i)_{i\in[n]}$ and function description F_λ. The challenger runs $\hat{m} \leftarrow \mathsf{Dec}(\mathsf{dk}_F, (\mathsf{ct}_i)_{i\in[n]})$ and returns $\hat{m}$ to $\mathcal{A}$. The set of decryption queries $\mathcal{Q}_{\mathsf{Dec}}$ is updated to include $\big((\mathsf{ct}_i)_{i\in[n]}, F_\lambda\big)$.*

6. *(Finalize) The adversary $\mathcal{A}$ outputs a guess $\hat{b}$. If the following conditions is satisfied and $\mathcal{Q}_{\mathsf{Dec}}$ contains* no *challenge components, the experiment outputs $\hat{b} \stackrel{?}{=} b$. Let $I \subset [n]$ be the set of corrupted indices, for $b \in \{0,1\}$ we define $\mathbf{X}^{(b)} := \{x^{(b)}_{1,j}, \ldots, x^{(b)}_{n,j}\}_{j=1}^{q}$ to be the q queried challenges*
   (a) *The pair $\mathbf{X}^{(0)}, \mathbf{X}^{(1)}$ satisfies that for all F queried by $\mathcal{A}$, all $I' = \{i_1, \ldots, i_t\} \subseteq I$, all $\{x'_{i_1}, \ldots, x'_{i_t}\}$, all $j_1, \ldots, j_{n-t} \in [q]$ we have*

   $$F\left(\mathsf{order}\left(x^{(0)}_{i_1,j_1}, \ldots, x^{(0)}_{i_{n-t},j_{n-t}}, x'_{i_1}, \ldots, x'_{i_t}\right)\right) = F\left(\mathsf{order}\left(x^{(1)}_{i_1,j_1}, \ldots, x^{(1)}_{i_{n-t},j_{n-t}}, x'_{i_1}, \ldots, x'_{i_t}\right)\right)$$

   (b) *The set $\{F\}$ queried by $\mathcal{A}$ satisfies that for all $\mathbf{X}^{(0)}, \mathbf{X}^{(1)}$ challenges, all $I' = \{i_1, \ldots, i_t\} \subseteq I$, all $\{x'_{i_1}, \ldots, x'_{i_t}\}$, all $j_1, \ldots, j_{n-t} \in [q]$ we have*

   $$F\left(\mathsf{order}\left(x^{(0)}_{i_1,j_1}, \ldots, x^{(0)}_{i_{n-t},j_{n-t}}, x'_{i_1}, \ldots, x'_{i_t}\right)\right) = F\left(\mathsf{order}\left(x^{(1)}_{i_1,j_1}, \ldots, x^{(1)}_{i_{n-t},j_{n-t}}, x'_{i_1}, \ldots, x'_{i_t}\right)\right)$$

   such that the ℓ-input receives its correspond value by the permutation $\mathsf{order}(\cdot)$*. Intuitively the set of inputs $\{x'_{i_1}, \ldots, x'_{i_t}\}$ represents whatever the adversary can put into the (subsets of) corrupted slots, and syntactically we use the permutation* $\mathsf{order}(\cdot)$ *to map values to their correct ordered arguments of the function (*e.g. *input value x'_{i_1} to argument k if $i_1 = k \in \mathbb{N}$). Otherwise, the experiment outputs* 0*.*

Last but not least, we give below the definition of *tag-based* CCA-security for MIFE. In the main body, to boostrap our IND-CPA secure MIFE construction, we first go through a tag-based CCA secure MIFE construction, then apply standard technique using one-time signatures to obtain the IND-CCA secure MIFE construction.

Definition 48 (Tag-based CCA security). *An MIFE scheme $\mathcal{E}$ with respect to a class of functions $\mathcal{F}$ is* secure *if for every* ppt *adversary $\mathcal{A}$, the following probability is negligible in λ:*

$$\mathsf{Adv}^{\mathsf{mife}}_{\mathcal{F},\mathcal{A}}(1^\lambda) := \left|\Pr[\mathsf{Expr}^{\mathsf{mife}}_{\mathcal{F},\mathcal{A}}(1^\lambda) = 1] - \frac{1}{2}\right|$$

where the experiment $\mathsf{Expr}^{\mathsf{mife}}_{\mathcal{F},\mathcal{A}}(1^\lambda)$ is defined as follows:

1. *The challenger runs $\mathsf{Setup}(1^\lambda, 1^n)$ to obtain $(\mathsf{pp}, \mathsf{msk}, (\mathsf{ek}_i)_{i\in[n]})$ and outputs pp to $\mathcal{A}$. In the following the adversary $\mathcal{A}$ can make queries adaptively in any order before Finalize.*
2. *(Key queries) The adversary $\mathcal{A}$ adaptively outputs a function description F_λ. The challenger runs $\mathsf{dk}_F \leftarrow \mathsf{Extract}(F_\lambda, \mathsf{msk})$ and returns dk_F to $\mathcal{A}$.*
3. *(Challenge) The adversary $\mathcal{A}$ outputs a query $(i, m_i^{(0)}, m_i^{(1)}, \mathsf{tag}^*)$ for some $i \in [n]$. The challenger chooses a bit $b \in \{0,1\}$ and encrypts $m_i^{(b)}$ to obtain $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, m_i^{(b)}, \mathsf{tag}^*)$. The ciphertext ct_i is returned to $\mathcal{A}$.*
4. *(Encryption) The adversary $\mathcal{A}$ outputs a query (i, m_i) for some $i \in [n]$. The challenger encrypts m_i to obtain $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, m_i, \mathsf{tag})$. The ciphertext ct_i is returned to $\mathcal{A}$.*
5. *(Decryption) The adversary $\mathcal{A}$ outputs at set of ciphertexts and tag $((\mathsf{ct}_i)_{i\in[n]}, \mathsf{tag}')$ and function description F_λ. The challenger runs $\hat{m} \leftarrow \mathsf{Dec}(\mathsf{dk}_F, (\mathsf{ct}_i)_{i\in[n]}, \mathsf{tag}')$ and returns $\hat{m}$ to $\mathcal{A}$. The set of decryption queries $\mathcal{Q}_{\mathsf{Dec}}$ is updated to include $\left((\mathsf{ct}_i)_{i\in[n]}, F_\lambda, \mathsf{tag}'\right)$.*
6. *(Finalize) The adversary $\mathcal{A}$ outputs a guess $\hat{b}$. If the following conditions is satisfied and $\mathcal{Q}_{\mathsf{Dec}}$ contains* no *challenge components and it holds that $\mathsf{tag}' \neq \mathsf{tag}^*$ for all tag' in $\mathcal{Q}_{\mathsf{Dec}}$, the experiment outputs $\hat{b} \stackrel{?}{=} b$. Let $I \subset [n]$ be the set of corrupted indices, for $b \in \{0,1\}$ we define $\mathbf{X}^{(b)} := \{x^{(b)}_{1,j}, \ldots, x^{(b)}_{n,j}\}_{j=1}^{q}$ to be the q queried challenges*

(a) The pair $\mathbf{X}^{(0)}, \mathbf{X}^{(1)}$ satisfies that for all F queried by $\mathcal{A}$, all $I' = \{i_1, \ldots, i_t\} \subseteq I$, all $\{x'_{i_1}, \ldots, x'_{i_t}\}$, all $j_1, \ldots, j_{n-t} \in [q]$ we have

$$\begin{aligned} &F\left(\mathsf{order}\left(x^{(0)}_{i_1,j_1}, \ldots, x^{(0)}_{i_{n-t},j_{n-t}}, x'_{i_1}, \ldots, x'_{i_t}\right)\right) \\ &\quad = F\left(\mathsf{order}\left(x^{(1)}_{i_1,j_1}, \ldots, x^{(1)}_{i_{n-t},j_{n-t}}, x'_{i_1}, \ldots, x'_{i_t}\right)\right) \end{aligned}$$

(b) The set $\{F\}$ queried by $\mathcal{A}$ satisfies that for all $\mathbf{X}^{(0)}, \mathbf{X}^{(1)}$ challenges, all $I' = \{i_1, \ldots, i_t\} \subseteq I$, all $\{x'_{i_1}, \ldots, x'_{i_t}\}$, all $j_1, \ldots, j_{n-t} \in [q]$ we have

$$\begin{aligned} &F\left(\mathsf{order}\left(x^{(0)}_{i_1,j_1}, \ldots, x^{(0)}_{i_{n-t},j_{n-t}}, x'_{i_1}, \ldots, x'_{i_t}\right)\right) \\ &\quad = F\left(\mathsf{order}\left(x^{(1)}_{i_1,j_1}, \ldots, x^{(1)}_{i_{n-t},j_{n-t}}, x'_{i_1}, \ldots, x'_{i_t}\right)\right) \end{aligned}$$

*such that the ℓ-input receives its correspond value by the permutation $\mathsf{order}(\cdot)$. Intuitively the set of inputs $\{x'_{i_1}, \ldots, x'_{i_t}\}$ represents whatever the adversary can put into the (subsets of) corrupted slots, and syntactically we use the permutation $\mathsf{order}(\cdot)$ to map values to their correct ordered arguments of the function (*e.g.* input value x'_{i_1} to argument k if $i_1 = k \in \mathbb{N}$). Otherwise, the experiment outputs 0.*

A.5 Weaker Notions of our IND-CCA Security

Weaker notions. We can relax Definition 29 to obtain weaker security notions.

- Similar to the *public key* encryption, extended to (single client) FE as in [18], we can define then weaker notion of *CCA1* security that allows the adversary to query to **Dec** only *before* the challenge phase. We denote the corresponding experiment by $\mathsf{Expr}^{\mathsf{w\text{-}rep\text{-}cca1}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$.
- In previous works, one can consider a weaker notion of security for MCFE in which either all or none of honest components in the challenge are queried. In this case, we say that the MCFE scheme is secure only against *complete* queries and add the following condition to the admissibility:
  1. There exist a tag tag and $i, j \in \mathcal{H}$ such that $i \neq j$, there exists a query $(i, x_i^{(0)}, x_i^{(1)}, (\mathsf{tag}, *))$ to **LoR** but there exist no query $(j, x_j^{(0)}, x_j^{(1)}, (\mathsf{tag}, *))$ to **LoR**.

  We denote the corresponding experiment with this weaker notion in admissibility, *i.e.* which is called *pos*-security in the literature, by $\mathsf{Expr}^{\mathsf{pos\text{-}w\text{-}rep\text{-}cca}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$.
- One can also recover the original security notion from [30] by imposing the *same* challenge components for corrupted $i \in \mathcal{C}$:
  1. There exists $i \in \mathcal{C}$ such that $x_i^{(0)} \neq x_i^{(1)}$.

  We denote the corresponding experiment with this weaker notion in admissibility by $\mathsf{Expr}^{\mathsf{wk\text{-}xxx\text{-}w\text{-}rep\text{-}cca}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$, where $\mathsf{xxx} \in \{\mathsf{sel}, \mathsf{pos}, \mathsf{1chal}\}$.
- In a more relaxed notion, the scheme $\mathcal{E}$ is *selectively IND-secure* if the following probability is negligible

$$\mathsf{Adv}^{\mathsf{sel\text{-}w\text{-}rep\text{-}cca}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) := \left|\Pr[\mathsf{Expr}^{\mathsf{sel\text{-}w\text{-}rep\text{-}cca}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) = 1] - \frac{1}{2}\right| .$$

  We also define a notion of security where only one challenge tag tag^* is allowed. That is, the scheme $\mathcal{E}$ is *one-time IND-secure* if the following probability is negligible

$$\mathsf{Adv}^{\mathsf{w\text{-}rep\text{-}1chal\text{-}cca}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) := \left|\Pr[\mathsf{Expr}^{\mathsf{w\text{-}rep\text{-}1chal\text{-}cca}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) = 1] - \frac{1}{2}\right| .$$

  The corresponding experiments can be found in Figure 3.

$\underline{\mathsf{Setup}(1^\lambda)}$: Choose $n+1$ pairs of dual orthogonal bases $(\mathbf{H}_i, \mathbf{H}_i^*, \mathbf{B}_i, \mathbf{B}_i^*)$ for $i \in [n]$ and $(\mathbf{F}, \mathbf{F}^*, \mathbf{G}, \mathbf{G}^*)$,$(\mathbf{H}_i, \mathbf{H}_i^*)$ is a pair of dual bases for $(\mathbb{G}_1^{2N+4}, \mathbb{G}_2^{2N+4})$, $(\mathbf{B}_i, \mathbf{B}_i^*)$ is a pair of dual bases for $(\mathbb{G}_1^{N+4}, \mathbb{G}_2^{N+4})$, $(\mathbf{F}, \mathbf{F}^*)$ is a pair of dual bases for $(\mathbb{G}_1^{2N+6}, \mathbb{G}_2^{2N+6})$, $(\mathbf{G}, \mathbf{G}^*)$ is a pair of dual bases for $(\mathbb{G}_1^{2N+6}, \mathbb{G}_2^{2N+6})$ [a]. Sample $\mu \xleftarrow{\$} \mathbb{Z}_q^*, \mathbf{S}, \mathbf{U}, \xleftarrow{\$} \prod_{i=1}^n (\mathbb{Z}_q^*)^N$ and write $\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_n)$, $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_n)$. Perform an n-out-of-n secret sharing on 1, that is, choose $p_i \in \mathbb{Z}_q$ such that $1 = p_1 + \cdots + p_n$. Then, for each $i \in [n]$, sample N random values $\theta_{i,k} \xleftarrow{\$} \mathbb{Z}_q$. Output the master secret key and the encryption keys as

$$\begin{cases} \mathsf{msk} := \Big(\mathbf{S},\ \mathbf{U},\ (\theta_{i,k})_{i\in[N],k\in[N]}, (\mathbf{b}^*_{i,k})_{k\in[N+2]},\ \mathbf{f}_1^*,\ \mathbf{f}_2^*,\ \mathbf{f}_3^*, \\ \qquad \mathbf{g}_1^*,\ \mathbf{g}_2^*,\ \mathbf{g}_3^*,\ (\mathbf{h}^*_{i,1}, \mathbf{h}^*_{i,2}, \mathbf{h}^*_{i,3}, (\mathbf{h}^*_{i,N+3+k})_{k=1}^N)_{i\in[n]}\Big) \\ \mathsf{ek}_i := \Big(\mathbf{s}_i,\ \mathbf{u}_i, (B_i^{(k)})_{k\in[N+2]}, \mathbf{b}_{i,N+3},\ \mathbf{f}_1,\ \mathbf{f}_2,\ \mathbf{f}_3, \\ \qquad \mathbf{g}_1,\ \mathbf{g}_2,\ \mathbf{g}_3,\ p_i \cdot H_i^{(1)},\ p_i \cdot H_i^{(2)},\ \mathbf{h}_{i,3},\ (\theta_{i,k}\mathbf{h}_{i,N+3+k})_{k=1}^N\Big) \end{cases}$$

where $H_i^{(k)}, B_i^{(k)}$ denotes the k-th row of H_i, B_i respectively.

$\underline{\mathsf{Extract}(\mathsf{msk}, (\mathbf{y}_i)_{i\in[n]} \in \prod_{i=1}^n \mathbb{Z}_q^N)}$: For each $i \in [n]$, each $k \in [N]$, sample $d_{\mathbb{A},i,k} \xleftarrow{\$} \mathbb{Z}_q$ such that $\sum_{i=1}^n \sum_{k=1}^N \theta_{i,k} d_{\mathbb{A},i,k} = 0$. For each $i \in [n]$, compute

$$\mathbf{m}_i := \left(\mathbf{y}_i, \sum_{i=1}^n a_{i,0}, \mathsf{rnd}_i, 0, 0\right)_{\mathbf{B}_i^*} \quad ; \widetilde{\mathbf{m}}_{i,0} := (\tilde{\pi}_{i,0} \cdot (n,1),\ a_{i,0},\ 0^N,\ 0,\ 0^N,\ 0,\ 0)_{\mathbf{G}^*}$$

$$\mathbf{k}_{i,0} := (\pi_{i,0} \cdot (n,1),\ a_{i,0} \cdot z,\ 0^N,\ 0,\ 0^N,\ 0,\ 0)_{\mathbf{F}^*}$$

$$\mathbf{k}_{i,\mathsf{ipfe}} := \left(\sum_{i=1}^n \langle \mathbf{s}_i, \mathbf{y}_i\rangle, \sum_{i=1}^n \langle \mathbf{u}_i, \mathbf{y}_i\rangle,\ a_{i,0} \cdot z,\ 0^N,\ (d_{\mathbb{A},i,k})_{k=1}^N, \mathsf{rnd}_{i,\mathsf{ipfe}}\right)_{\mathbf{H}_i^*}$$

where $z, \pi_{i,0}, \mathsf{rnd}_i, \mathsf{rnd}_{i,\mathsf{ipfe}} \xleftarrow{\$} \mathbb{Z}_q$. Output $\mathsf{dk}_{\mathbb{A},\mathbf{y}} := \Big(\mathbf{k}_{i,0}, \widetilde{\mathbf{m}}_{i,0}, (\mathbf{m}_i, \mathbf{k}_{i,\mathsf{ipfe}})_{i\in[n]}\Big)$.

$\underline{\mathsf{Enc}(\mathsf{ek}_i, \mathbf{x}_i \in \mathbb{Z}_q^N, z_i := \mathsf{tag})}$: Parse

$$\mathsf{ek}_i := \Big(\mathbf{s}_i,\ \mathbf{u}_i, (B_i^{(k)})_{k\in[N+2]}, \mathbf{b}_{i,N+3}, \mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3, p_i \cdot H_i^{(1)},\ p_i \cdot H_i^{(2)},\ \mathbf{h}_{i,3},\ (\theta_{i,k}\mathbf{h}_{i,N+3+k})_{k=1}^N\Big)$$

and $\mathsf{S}_i \subseteq \mathsf{Att} \subseteq \mathbb{Z}_q$ as the set of attributes, compute $\mathsf{H}(\mathsf{tag}) \to (\llbracket \omega \rrbracket_1\ ,\ \llbracket \omega' \rrbracket_1) \in \mathbb{G}_1^2$. Use $p_i H_i^{(1)}$ and $p_i H_i^{(2)}$ to compute

$$p_i H_i^{(1)} \cdot \llbracket \omega \rrbracket_1 + p_i H_i^{(2)} \cdot \llbracket \omega' \rrbracket_1 = p_i \cdot \left(\omega H_i^{(1)} \cdot g_1 + \omega' H_i^{(2)} \cdot g_1\right) = p_i \cdot (\omega \mathbf{h}_{i,1} + \omega' \mathbf{h}_{i,2})\ .$$

Compute

$$\widetilde{\mathbf{t}}_{i,0} = \widetilde{\sigma}_{i,0} \cdot (\mathbf{g}_1 - n \cdot \mathbf{g}_2) + \left(\sum_i \nu_i\right) \cdot \mathbf{g}_3 = (\widetilde{\sigma}_{i,0} \cdot (1,-n),\ \sum_i \nu_i,\ 0^N,\ 0,\ 0^N,\ 0,\ 0)_{\mathbf{G}}$$

$$\mathbf{c}_{i,0} = \sigma_{i,0} \cdot \mathbf{f}_1 - n \cdot \sigma_{i,0} \cdot \mathbf{f}_2 + \psi_i \cdot \mathbf{f}_3 = (\sigma_{i,0} \cdot (1,-n),\ \psi_i,\ 0^N,\ 0,\ 0^N,\ 0,\ 0)_{\mathbf{F}}$$

where $\widetilde{\sigma}_{i,0}, \sigma_{i,0} \xleftarrow{\$} \mathbb{Z}_q$. Finally, compute

$$\begin{aligned} \mathbf{t}_i &:= \sum_{k\in[N]} \left(\llbracket \omega \rrbracket_1 \cdot \mathbf{s}_i[k] \cdot B_i^{(k)} + \llbracket \omega' \rrbracket_1 \cdot \mathbf{u}_i[k] \cdot B_i^{(k)} + \llbracket \mathbf{x}_i[k] \rrbracket_1\right) + \nu_i \cdot \mathbf{b}_{i,N+1} + \rho_i \cdot \mathbf{b}_{i,N+3} \\ &= (\omega \cdot \mathbf{s}_i + \omega' \cdot \mathbf{u}_i + \mathbf{x}_i, \nu_i, 0, \rho_i)_{\mathbf{B}_i} \end{aligned}$$

$$\mathbf{c}_{i,\mathsf{ipfe}} := p_i \cdot (\omega \cdot \mathbf{h}_{i,1} + \omega' \cdot \mathbf{h}_{i,2}) + \psi_i \cdot \mathbf{h}_{i,3} + \sum_{k=1}^N \theta_{i,k}\mathbf{h}_{i,N+3+k} = (\omega p_i,\ \omega' p_i,\ \psi_i,\ 0^N,\ (\theta_{i,k})_{k=1}^N,\ 0)_{\mathbf{H}_i}$$

and output $\mathsf{ct}_{\mathsf{tag},i} := \Big(\mathbf{c}_{i,0}, \widetilde{\mathbf{t}}_{i,0}, \mathbf{t}_i, \mathbf{c}_{i,\mathsf{ipfe}}\Big)$.

$\underline{\mathsf{Dec}(\mathsf{dk}_{\mathbb{A},\mathbf{y}}, \mathbf{c} := (\mathsf{ct}_{\mathsf{tag},i}))}$: Parse

$$\mathsf{ct}_{\mathsf{tag},i} = \Big(\mathbf{c}_{i,0}, \widetilde{\mathbf{t}}_{i,0}, \mathbf{t}_i, \mathbf{c}_{i,\mathsf{ipfe}}\Big) \quad \text{and} \quad \mathsf{dk}_{\mathbb{A},\mathbf{y}} := \Big(\mathbf{k}_{i,0}, \widetilde{\mathbf{m}}_{i,0}, (\mathbf{m}_i, \mathbf{k}_{i,\mathsf{ipfe}})_{i\in[n]}\Big)\ .$$

and perform Algorithm in Figure 7. Finally, compute the discrete logarithm and output the small value $\mathsf{out} \in [-nNB^2, nNB^2] \subsetneq \mathbb{Z}_q$ [b].

[a] We denote the basis changing matrices for $(\mathbf{F}, \mathbf{F}^*), (\mathbf{B}_i, \mathbf{B}_i^*), (\mathbf{H}_i, \mathbf{H}_i^*)$ as $(F, F' := (F^{-1})^\top), (B_i, B_i' := (B_i^{-1})^\top), (H_i, H_i' := (H_i^{-1})^\top)$ respectively (see the appendix A.3 for basis changes in DPVS).

[b] we represent $\mathbb{Z}_q$ as the ring of integers with addition and multiplication modulo q, containing the representatives in the interval $(-q/2,\ q/2)$.

Fig. 6: The CPA-secure construction $\mathsf{MCFE}^{\mathsf{cpa}}$ for multi-client IPFE to be instantiated in Section 4.2.

Input: $\mathsf{ct}_{\mathsf{tag},i} = \left(\mathbf{c}_{i,0}, \widetilde{\mathbf{t}}_{i,0}, \mathbf{t}_i, \mathbf{c}_{i,\mathsf{ipfe}}\right)$ and $\mathsf{dk}_{\mathbb{A},\mathbf{y}} := \left(\mathbf{k}_{i,0}, \widetilde{\mathbf{m}}_{i,0}, (\mathbf{m}_i, \mathbf{k}_{i,\mathsf{ipfe}})_{i\in[n]}\right)$

1. For each i compute

$$X_i = \widetilde{\mathbf{t}}_{i,0} \times \widetilde{\mathbf{m}}_{i,0} = \left[\!\!\left[(\sum_i \nu_i) \cdot a_{i,0}\right]\!\!\right]_{\mathsf{t}}$$
$$Y_i = \mathbf{c}_{i,0} \times \mathbf{k}_{i,0} = [\![\psi_i \cdot a_{i,0} \cdot z]\!]_{\mathsf{t}}$$

   and in the end summing all X_i to obtain $\mathsf{mask} = \sum_i X_i = [\![(\sum_i \nu_i) \cdot (\sum_i a_{i,0})]\!]_{\mathsf{t}}$
2. Compute

$$W = \sum_i \mathbf{t}_i \times \mathbf{m}_i = \left[\!\!\left[\sum_i \left(\omega \cdot \langle \mathbf{s}_i, \mathbf{y}_i\rangle + \omega' \cdot \langle \mathbf{u}_i, \mathbf{y}_i\rangle + \langle \mathbf{x}_i, \mathbf{y}_i\rangle\right) + (\sum_i \nu_i) \cdot (\sum_i a_{i,0})\right]\!\!\right]_{\mathsf{t}}$$

   as well as $Z = \sum_i \left(\mathbf{c}_{i,\mathsf{ipfe}} \times \mathbf{k}_{i,\mathsf{ipfe}} - Y_i\right) = [\![\omega \cdot \sum_i \langle \mathbf{s}_i, \mathbf{y}_i\rangle + \omega' \cdot \sum_i \langle \mathbf{u}_i, \mathbf{y}_i\rangle]\!]_{\mathsf{t}}$ thanks to $\sum_{i=1}^{n}\sum_{k=1}^{N} \theta_{i,k} d_{\mathbb{A},i,k} = 0$ and $\sum_i p_i = 1$.
3. Finally, compute $\mathsf{out} = W - Z - \mathsf{mask} = [\![\sum_i \langle \mathbf{x}_i, \mathbf{y}_i\rangle]\!]_{\mathsf{t}}$ and then a discrete log of out in base g_{t} to obtain $\sum_i \langle \mathbf{x}_i, \mathbf{y}_i\rangle$.

Fig. 7: The final computation of decryption for the MCFE in Figure 6, whose *correctness* can be verified according to construction.

B MCFE for Instantiations - IND-CPA Secure MCFE from [67]

We recall the CPA-secure construction $\mathsf{MCFE}^{\mathsf{cpa}}$ to be instantiated in Section 4.2 in Figure 6.

C The NIZK for our MCFE in Section 4.2

Step 1: The Σ-protocol. We first give a Σ-protocol $\Sigma_{\mathsf{cpa,ee}}$ in Figure 8 for proving the parts that relate to the $\mathsf{MCFE}^{\mathsf{cpa}}$ and EECom, with respect to $(\mathsf{ct}, c^{\mathsf{ee},j})$ of the relation $\mathsf{CtMsgRel}^{\mathsf{sk}}$. For the sake of a modular presentation, we rewrite the witness of relation $\mathsf{CtMsgRel}^{\mathsf{sk}}$ as:

$$\begin{aligned}\mathsf{CtMsgRel}^{\mathsf{sk}} = \{((\mathsf{ct}, c^{\mathsf{ee},j}, c_{\mathsf{ek}}),\quad & \omega := \left((wit^{\mathsf{cpa}} \overset{def}{=} (\mathsf{ek}, r), wit^{\mathsf{ee}} \overset{def}{=} (d^{\mathsf{ee}}, m), d_{\mathsf{ek}}\right))) \ : \\ & \mathsf{ct} = \mathsf{Enc}^{\mathsf{cpa}}(\mathsf{ek}, m; r) \\ \wedge\ & \mathsf{EECom.Verify}(\mathsf{pp}^{\mathsf{ee}}, c^{\mathsf{ee},j}, m, d_{\mathsf{ee}}) = 1 \\ \wedge\ & \mathsf{Com.Verify}(\mathsf{pp}^{\mathsf{com}}, c_{\mathsf{ek}}, \mathsf{ek}, d_{\mathsf{ek}}) = 1\}\ .\end{aligned}$$

When instantiating for the concrete ciphertexts and commitments of the MCFE scheme from Figure 6, the witness and statement are indexed by $i \in [n]$. The Σ-protocol involves $(\mathsf{ct}_i, c_i^{\mathsf{ee},j})$ for the part relating to the IND-CPA underlying and the extractable-equivocable commitment. We write the witness as $(wit_i^{\mathsf{cpa}}, wit_i^{\mathsf{ee}})$:

$$wit_i^{\mathsf{cpa}} \overset{def}{=} wit^{\mathsf{cpa}}(\widetilde{\mathbf{t}}, i), wit^{\mathsf{cpa}}(\mathbf{c}, i), (wit^{\mathsf{cpa}}(\mathbf{t}, i, k))_{k\in[N]}, wit^{\mathsf{cpa}}(\mathbf{c}, i, \mathsf{ipfe})$$

from Equation (14),Equation (15),Equation (16),Equation (17) and

$$wit_i^{\mathsf{ee}} \overset{def}{=} (wit^{\mathsf{ee}}(a, i, l, j))_{l,j}, (wit^{\mathsf{ee}}(b, i, l, j))_{l\in[L], j\in\{0,1\}}\quad ,$$

from Equation (18),Equation (19). In the above we define $M^{(i)} = (M_j^{(i)})_j \overset{def}{=} \mathsf{bin}(x_i, z_i) \in \{0,1\}^L, L \overset{def}{=} \lceil \log x_i \rceil + \lceil \log z_i \rceil$. The Σ-protocol $\Sigma_{\mathsf{cpa,ee}}$ is executed by each client $i \in [n]$ in order to prove that a witness $(wit_i^{\mathsf{cpa}} \overset{def}{=} (\mathsf{ek}, r), wit_i^{\mathsf{ee}} \overset{def}{=} (d^{\mathsf{ee}}, m))$ is in relation with a statement $(\mathsf{ct}_i, c_i^{\mathsf{ee},j})$, where:

$$\mathsf{ct}_i = \mathsf{Enc}^{\mathsf{cpa}}(\mathsf{ek}_i, m_i; r_i) \quad \wedge \quad \mathsf{EECom.Verify}(\mathsf{pp}^{\mathsf{ee}}, c_i^{\mathsf{ee},j}, m_i, d_{\mathsf{ee},i}) = 1\ . \tag{13}$$

The encryption $\mathsf{Enc}^{\mathsf{cpa}}$ is the MCFE encryption scheme (see Fig. 6), and the commitment EECom is the extractable-equivocable commitment scheme (see Fig. 1). Our Σ-protocol $\Sigma_{\mathsf{cpa,ee}}$ is described in Figure 8 in Appendix D.1 with a full proof of security by Lemma 49.

Lemma 49. *The Σ-protocol protocol $\Sigma_{\mathsf{cpa,ee}}$ in Figure 8 is* correct, HVZK, special sound *with* high min-entropy, *for proving witness* $(wit^{\mathsf{cpa}} \overset{def}{=} (\mathsf{ek}, r), wit^{\mathsf{ee}} \overset{def}{=} (d^{\mathsf{ee}}, m))$ *of statement* $(\mathsf{ct}_i, c_i^{\mathsf{ee},j})$ *with respect to Equation* (13) *in the relation* $\mathsf{CtMsgRel}^{\mathsf{sk}}$.

We then apply the Fiat-Shamir transform (Theorem 25) to the Σ-protocol in Figure 8. We recall that thanks to a result by [40] on the Fiat-Shamir transform for our 2-special soundness Σ-protocol, we achieve further simulation soundness for the resultant NIZK proof system.

Corollary 50. *Let* $\Sigma_{\mathsf{cpa,ee}}$ *be the* Σ*-protocol in Figure* 8*. Let* $\Pi^{ss}_{\mathsf{cpa,ee}} \overset{def}{=} FS[\Sigma_{\mathsf{cpa,ee}}]$ *be the result after applying the Fiat-Shamir transform to* $\Sigma_{\mathsf{cpa,ee}}$*. Then* $\Pi^{ss}_{\mathsf{cpa,ee}}$ *is a NIZK proof system for proving witness* $(wit_i^{\mathsf{cpa}}, wit_i^{\mathsf{ee}})$ *of statement* $(\mathsf{ct}_i, c_i^{\mathsf{ee},j})$*, that is* correct *and satisfies* simulation soundness*, with respect to the relation in* (13)*.*

Step 2: Combine with the Groth-Sahai NIZK, made Simulation-Sound. We only call implicitly the Groth-Sahai NIZK verifification, in order to convey the main parts of our protocol. Opening, or verifying the commitment of Groth-Sahai NIZK, can be done using the opening d together with the commitment key $\mathsf{pp}^{\mathsf{com}} = (\overrightarrow{\mathcal{U}}, \overrightarrow{\mathcal{V}}, \iota_1(\cdot), \iota_2(\cdot), \iota_1'(\cdot), \iota_2'(\cdot))$. It is important to note that the zero-knowledge property of the Groth-Sahai NIZK is obtained in the bilinear group setting under SXDH, as explained in [49, Sect. 7.1]. However, it is known that the Groth-Sahai NIZK is malleable and thus not simulation-sound, as necessarily required in our instantiation. Before proceeding, we apply techniques from [48, 24, 50, 51] to Groth-Sahai NIZK to make it simulation-sound and get $\Pi^{\mathsf{ss}}_{\mathsf{gs}}$ that is employed in the following[21].

The result of our instantiation for the NIZK proof system $\Pi^{\mathsf{ss}}_{\mathsf{gs,cpa,ee}}$ is obtained by by integrating the $\Pi^{\mathsf{ss}}_{\mathsf{gs}}$ with $\Pi^{\mathsf{ss}}_{\mathsf{cpa,ee}}$ from Corollary 50. The main steps of $\Pi^{\mathsf{ss}}_{\mathsf{gs,cpa,ee}}$ are given below: we index the statement and witness by $i \in [n]$ for to put the NIZK directly into context, so that hopefully the reading is facilitated, *i.e.* in the MCFE as from Section 4.1 this NIZK is applied for every i by each client i.

The NIZK $\Pi^{\mathsf{ss}}_{\mathsf{gs,cpa,ee}}$ for $\mathsf{CtMsgRel}^{\mathsf{sk}}$

(We index the statement and witness by $i \in [n]$ for the context of using this NIZK later in the MCFE)

Prover$((\mathsf{ct}_i, c_i^{\mathsf{ee},j}, c_{\mathsf{ek},i}),\quad (wit_i^{\mathsf{cpa}} \overset{def}{=} (\mathsf{ek}_i, r_i), wit_i^{\mathsf{ee}} \overset{def}{=} (d_i^{\mathsf{ee}}, m_i), d_{\mathsf{ek},i}))$**:** Use $\Pi^{\mathsf{ss}}_{\mathsf{cpa,ee}}$ from Corollary 50 to prove the knowledge of wit^{cpa} and wit^{ee} such that

$$\mathsf{ct}_i = \mathsf{Enc}^{\mathsf{cpa}}(\mathsf{ek}_i, m_i; r_i) \quad \wedge \quad \mathsf{EECom.Verify}(\mathsf{pp}^{\mathsf{ee}}, c_i^{\mathsf{ee},j}, m_i, d_{\mathsf{ee},i}) = 1 \ .$$

The obtained proof is denoted $\pi_i^{\mathsf{cpa,ee}}$. Then, use the $\Pi^{\mathsf{ss}}_{\mathsf{gs}}$ and the knowledge of d_{ek} to prove that ek_i satisfies

$$\mathsf{Com.Verify}(\mathsf{pp}^{\mathsf{com}}, c_{\mathsf{ek},i}, \mathsf{ek}_i, d_{\mathsf{ek},i}) = 1 \ .$$

We emphasize here that it is important that the same ek_i is used in the proof of $\Pi^{\mathsf{ss}}_{\mathsf{cpa,ee}}$ and the $\Pi^{\mathsf{ss}}_{\mathsf{gs}}$. The obtained proof is denoted π_i^{gs}. Finally, the prover sends $(\pi_i^{\mathsf{cpa,ee}}, \pi_i^{\mathsf{gs}})$ to the verifier.

Verifier$(\mathsf{ct}_i, c_i^{\mathsf{ee},j}, c_{\mathsf{ek},i})$**:** Upon receiving $(\pi_i^{\mathsf{cpa,ee}}, \pi_i^{\mathsf{gs}})$ accept if and only if both $\pi_i^{\mathsf{cpa,ee}}$ and π_i^{gs} are accepted by the verification of $\Pi^{\mathsf{ss}}_{\mathsf{cpa,ee}}$ and the verification of the $\Pi^{\mathsf{ss}}_{\mathsf{gs}}$, respectively. Following the above notation, the Groth-Sahai checks can be done in batch *as per* [20, Sect. 5]. For completeness the main pairing computations are recalled in Appendix E.1.

Theorem 51. *The proof system* $\Pi^{ss}_{\mathsf{gs,cpa,ee}}$ *is a NIZK proof system for the relation* $\mathsf{CtMsgRel}^{\mathsf{sk}}$*, satisfying* completeness, simulation soundness, *and* zero-knowledge*.*

Proof (Sketch). For *completeness*, we can conclude based on the completeness of the underlying proof systems. For *zero-knowledge*,

[21] If we aim for *one-time* simulation soundness, that is, the adversary sees only one simulated proof in the soundness game, there are more economical constructions that result in shorter proofs from [53, 52, 60, 54] and their follow-ups. However, as already detailed in the proof of Theorem 33, due to the authorized repetitions on fixed (i, tag) that lead to repetitive proofs on different inputs $(x_i^{(j)}, z_i^{(j)})$, one-time simulation soundness is not sufficient.

- First of all the proof system $\mathit{\Pi}^{\mathsf{ss}}_{\mathsf{cpa,ee}}$ is zero-knowledge following the Fiat-Shamir transform in Corollary 50.
- Then, thanks to the fact that the Groth-Sahai NIZK is composable zero-knowledge [49, Theorem 5], which preserves under techniques from [48, 24, 50, 51] to make it simulation-sound, anding π_i^{gs} on top of $\pi_i^{\mathsf{cpa,ee}}$ does not leak more information about ek_i in $d_{\mathsf{ek},i}$. In particular this does not leak more information about the secret key ek_i that is common to both proofs.

For *simulation soundness*, against a PPT adversary $\mathcal{A}$ that plays against the simulation soundness of $\mathit{\Pi}^{\mathsf{ss}}_{\mathsf{gs,cpa,ee}}$, we reduce either to break the simulation soundness of $\mathit{\Pi}^{\mathsf{ss}}_{\mathsf{cpa,ee}}$ or the simulation soundness of $\mathit{\Pi}^{\mathsf{ss}}_{\mathsf{gs}}$.

On one hand, we apply the simulation soundness that comes from the proof system $\mathit{\Pi}^{\mathsf{ss}}_{\mathsf{cpa,ee}}$ with respect to the (potentially false) statement $(\mathsf{ct}_i, c_i^{\mathsf{ee}})$: to respond to a simulation query for $\mathit{\Pi}^{\mathsf{ss}}_{\mathsf{gs,cpa,ee}}$, we ask to the soundness challenger of $\mathit{\Pi}^{\mathsf{ss}}_{\mathsf{cpa,ee}}$ to simulate the proof $\pi_i^{\mathsf{cpa,ee}}$. This is under the simulated set up $(\widetilde{\mathsf{crs}}^{\mathsf{ss}}_{\mathsf{cpa,ee}}, \mathsf{td}^{\mathsf{ss}}_{\mathsf{cpa,ee}}) \leftarrow \mathsf{SimCRS}^{\mathsf{ss}}_{\mathsf{cpa,ee}}(1^\lambda)$ and we obtain from the challenger a simulated proof in the soundness game of $\mathit{\Pi}^{\mathsf{ss}}_{\mathsf{cpa,ee}}$. Using the simulated set up of $\mathit{\Pi}^{\mathsf{ss}}_{\mathsf{gs}}$, on the other hand, we can simulate the proof π_i^{gs} under the simulated set up $(\widetilde{\mathsf{crs}}^{\mathsf{ss}}_{\mathsf{com}}, \mathsf{td}^{\mathsf{ss}}_{\mathsf{com}}) \leftarrow \mathsf{SimCRS}^{\mathsf{ss}}_{\mathsf{com}}(1^\lambda)$.

One subtlety is an edge case where the witness $(wit_i^{\mathsf{cpa}} \overset{def}{=} (\mathsf{ek}_i, r_i), wit_i^{\mathsf{ee}} \overset{def}{=} (d_i^{\mathsf{ee}}, m_i), d_{\mathsf{ek},i})$ and $d_{\mathsf{ek},i}$ contains an opening $\widetilde{\mathsf{ek}}_i$ for $c_{\mathsf{ek},i}$ so that $\mathsf{ek}_i \neq \widetilde{\mathsf{ek}}_i$ are different. Unless the soundness of $\mathit{\Pi}^{\mathsf{ss}}_{\mathsf{gs}}$ or $\mathit{\Pi}^{\mathsf{ss}}_{\mathsf{cpa,ee}}$ is broken, this case can be excluded by the binding property of $c_{\mathsf{ek},i}$ that comes with $\mathit{\Pi}^{\mathsf{ss}}_{\mathsf{gs}}$, which *a priori* inherits the *perfect* binding guarantee from the commitment in Groth-Sahai NIZK. This is due to the fact that the perfect binding from Groth-Sahai commitments binds $\widetilde{\mathsf{ek}}_i$ to a solution of a multi-scalar equation, which also represents the encryption equation in $\mathbb{G}_1$ of $\mathsf{MCFE}^{\mathsf{cpa}}.\mathsf{Enc}$ using ek_i, proved by $\mathit{\Pi}^{\mathsf{ss}}_{\mathsf{cpa,ee}}$. Given fixed public parameters and thanks to linearity, the scalars and group elements of ek_i are the unique solution for

$$\mathsf{ct}_i = \begin{cases} \widetilde{\mathbf{t}}_{i,0} & = \widetilde{\sigma}_{i,0} \cdot (\mathbf{g}_1 - n \cdot \mathbf{g}_2) + \nu \cdot \mathbf{g}_3 \\ \mathbf{c}_{i,0} & = \sigma_{i,0} \cdot \mathbf{f}_1 - n \cdot \sigma_{i,0} \cdot \mathbf{f}_2 + \psi_i \cdot \mathbf{f}_3 \\ \mathbf{t}_i & = \sum_{k \in [N]} \left([\![\omega]\!]_1 \cdot \mathbf{s}_i[k] \cdot B_i^{(k)} + [\![\omega']\!]_1 \cdot \mathbf{u}_i[k] \cdot B_i^{(k)} + [\![\mathbf{x}_i[k]]\!]_1 \right) \\ & \quad + \nu_i \cdot \mathbf{b}_{i,N+1} + \rho_i \cdot \mathbf{b}_{i,N+3} \\ \mathbf{c}_{i,\mathsf{ipfe}} & = p_i H_i^{(1)} \cdot [\![\omega]\!]_1 + p_i H_i^{(2)} \cdot [\![\omega']\!]_1 + \psi_i \cdot \mathbf{h}_{i,3} + \sum_{k=1}^{N} \theta_{i,k} \mathbf{h}_{i,N+3+k} \end{cases},$$

and thus $\mathsf{ek}_i \neq \widetilde{\mathsf{ek}}_i$ give a contradiction[22]. □

D Σ-protocol and Fiat-Shamir Transform - Instantiations for Section 4.2

Statement and Witness. First of all, the commiment scheme $\mathsf{Com} = (\mathsf{Com.Setup}, \mathsf{Com.Commit}, \mathsf{Com.Verify})$ is instantiate by the commitment step in Groth-Sahai [49], which is recalled in Appendix E, then the NIZK to verify commitments of ek_i can be instatiated with Groth-Sahai, while the remaining discrete logarithm relations can be treated using Schnorr under Fiat-Shamir. The relation $\mathsf{CtMsgRel}^{\mathsf{sk}}$ is for proving with respect to each client i. More specifically, the statement $(\mathsf{ct}, c^{\mathsf{ee},j}, c_{\mathsf{ek}})$ of $\mathsf{CtMsgRel}^{\mathsf{sk}}$ is given by: for each $i \in [n]$ the statement for each part is

Statement $(\mathsf{ct}, c^{\mathsf{ee},j}, c_{\mathsf{ek}})$ of $\mathsf{CtMsgRel}^{\mathsf{sk}}$

[22] We essentially needs perfect binding, otherwise an unbounded adversary can include ek_i in the opening $d_{\mathsf{ek},i}$ to open to a commitment $c_{\mathsf{ek},i}$ of some different $\widetilde{\mathsf{ek}}_i$, making the argument of unique solution useless.

$\underline{\mathsf{MCFE}^{\mathsf{cpa}}}$:

$$\mathsf{ct}_i = \begin{cases} \widetilde{\mathbf{t}}_{i,0} & = \widetilde{\sigma}_{i,0} \cdot (\mathbf{g}_1 - n \cdot \mathbf{g}_2) + \nu \cdot \mathbf{g}_3 \\ \mathbf{c}_{i,0} & = \sigma_{i,0} \cdot \mathbf{f}_1 - n \cdot \sigma_{i,0} \cdot \mathbf{f}_2 + \psi_i \cdot \mathbf{f}_3 \\ \mathbf{t}_i & = \sum_{k \in [N]} \left([\![\omega]\!]_1 \cdot \mathbf{s}_i[k] \cdot B_i^{(k)} + [\![\omega']\!]_1 \cdot \mathbf{u}_i[k] \cdot B_i^{(k)} + [\![\mathbf{x}_i[k]]\!]_1 \right) \\ & \quad +\nu_i \cdot \mathbf{b}_{i,N+1} + \rho_i \cdot \mathbf{b}_{i,N+3} \\ \mathbf{c}_{i,\mathsf{ipfe}} & = p_i H_i^{(1)} \cdot [\![\omega]\!]_1 + p_i H_i^{(2)} \cdot [\![\omega']\!]_1 + \psi_i \cdot \mathbf{h}_{i,3} + \sum_{k=1}^{N} \theta_{i,k} \mathbf{h}_{i,N+3+k} \end{cases}$$

$\underline{\mathsf{EECom}}$ (Fig. 1): $U = g_1^a h_1^b, V = g_1^c h_1^d, W = g_1^e, T = g_2^t$

$$M^{(i)} = (M_j^{(i)})_j \overset{def}{=} \mathsf{bin}(x_i, z_i) \in \{0,1\}^L, L \overset{def}{=} \lceil \log x_i \rceil + \lceil \log z_i \rceil$$

$$c_i^{\mathsf{ee},j} = \begin{cases} \forall l \in [L], j \in \{0,1\} : \mathbf{a}_l^{(i)} \overset{def}{=} \left[\!\!\left[\mathsf{rnd}_{l, M_l^{(i)}} \right]\!\!\right]_2 + M_l^{(i)} \cdot T \in \mathbb{G}_2, d_{l,j} \overset{def}{=} [\![\mathsf{rnd}_{l,j}]\!]_1 \in \mathbb{G}_1 \\ \forall l \in [L], j \in \{0,1\} : \mathbf{b}_{l,j}^{(i)} = \\ \qquad (u_{l,j} = g_1^{s_{l,j}}, v_{l,j} = h_1^{s_{l,j}}, t_{l,j} = d_{l,j} + W^{s_{l,j}}, w_{l,j} = U^{s_{l,j}} V^{s_{l,j} \cdot \sigma}) \in \mathbb{G}_1^4 \end{cases}$$

$\underline{\mathsf{Com}}$ (Groth-Sahai to commit to ek_i's contents):

$$\mathsf{ek}_i \overset{def}{=} \Big(\mathbf{s}_i,\ \mathbf{u}_i, (B_i^{(k)})_{k \in [N+2]}, \mathbf{b}_{i,N+3},\ \mathbf{f}_1,\ \mathbf{f}_2,\ \mathbf{f}_3,$$
$$\mathbf{g}_1,\ \mathbf{g}_2,\ \mathbf{g}_3,\ p_i \cdot H_i^{(1)},\ p_i \cdot H_i^{(2)},\ \mathbf{h}_{i,3},\ (\theta_{i,k} \mathbf{h}_{i,N+3+k})_{k=1}^N \Big)$$

$$\overset{def}{=} \Big(\mathsf{sc}(\mathsf{ek}_i), \mathsf{grp}(\mathsf{ek}_i) \Big) \text{ where } \begin{cases} \mathsf{sc}(\mathsf{ek}_i) & \overset{def}{=} (\mathbf{s}_i, \mathbf{u}_i, (B_i^{(k)})_{k \in [N+2]}, p_i \cdot H_i^{(1)}, p_i \cdot H_i^{(2)}) \\ \mathsf{grp}(\mathsf{ek}_i) & \overset{def}{=} (\mathbf{b}_{i,N+3}, \mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3, \mathbf{h}_{i,3}, (\theta_{i,k} \mathbf{h}_{i,N+3+k})_{k=1}^N) \end{cases}$$

$$c_{\mathsf{ek},i} = \begin{cases} \forall j \in [\mathsf{len}(\mathsf{sc}(\mathsf{ek}_i))] : \mathbf{c}_{\mathsf{ek},\mathsf{sc},i,j} & \overset{def}{=} \iota_1'(\mathsf{sc}(\mathsf{ek}_i)[j]) \odot s_{i,j} \mathbf{u}_1 \in \mathbb{G}_1^2, s_{i,j} \overset{\$}{\leftarrow} \mathbb{Z}_q^* \\ \mathbf{d}_{\mathsf{ek},\mathsf{sc},i,j} & \overset{def}{=} \iota_2'(\mathsf{sc}(\mathsf{ek}_i)[j]) \odot s_{i,j} \mathbf{v}_1 \in \mathbb{G}_2^2, s_{i,j} \overset{\$}{\leftarrow} \mathbb{Z}_q^* \\ \forall j \in [\mathsf{len}(\mathsf{grp}(\mathsf{ek}_i))] : \mathbf{c}_{\mathsf{ek},\mathsf{grp},i,j} & \overset{def}{=} \iota_1(\mathsf{grp}(\mathsf{ek}_i)[j]) \odot s_{i,j,1} \mathbf{u}_1 \odot s_{i,j,2} \mathbf{u}_2 \in \mathbb{G}_1^2, s_{i,j,1}, s_{i,j,2} \overset{\$}{\leftarrow} \mathbb{Z}_q^* \end{cases}$$

in which $\sigma = \mathsf{H}^{\mathsf{cr}}(\mathsf{tag}, (u_{i,j}, v_{i,j}, e_{i,j})_{i,j})$ and $\mathsf{H}(\mathsf{tag}) = [\![(\omega, \omega')]\!]_1$. The commitment key of Groth-Sahai can be revisted in equations (25). We remark that we only use the commitments to scalars (item 3) and to group elements in $\mathbb{G}_1$ (item 1). The equations for which theses Groth-Sahai commitments will be proved upon are those that relate to $\mathsf{MCFE}^{\mathsf{cpa}}.\mathsf{Enc}$, *i.e.* the multi-scalar equations in (23):

$$\mathsf{ct}_i = \begin{cases} \widetilde{\mathbf{t}}_{i,0} & = \widetilde{\sigma}_{i,0} \cdot (\mathbf{g}_1 - n \cdot \mathbf{g}_2) + \nu \cdot \mathbf{g}_3 \\ \mathbf{c}_{i,0} & = \sigma_{i,0} \cdot \mathbf{f}_1 - n \cdot \sigma_{i,0} \cdot \mathbf{f}_2 + \psi_i \cdot \mathbf{f}_3 \\ \mathbf{t}_i & = \sum_{k \in [N]} \left([\![\omega]\!]_1 \cdot \mathbf{s}_i[k] \cdot B_i^{(k)} + [\![\omega']\!]_1 \cdot \mathbf{u}_i[k] \cdot B_i^{(k)} + [\![\mathbf{x}_i[k]]\!]_1 \right) \\ & \quad +\nu_i \cdot \mathbf{b}_{i,N+1} + \rho_i \cdot \mathbf{b}_{i,N+3} \\ \mathbf{c}_{i,\mathsf{ipfe}} & = p_i H_i^{(1)} \cdot [\![\omega]\!]_1 + p_i H_i^{(2)} \cdot [\![\omega']\!]_1 + \psi_i \cdot \mathbf{h}_{i,3} + \sum_{k=1}^{N} \theta_{i,k} \mathbf{h}_{i,N+3+k} \end{cases} .$$

This is important to ensure the correctness of the encryption keys ek_i both via what is computed in the ciphertext ct_i and the commitment $c_{\mathsf{ek},i}$.

The witness $\omega = (\mathsf{ek}, r, d^{\mathsf{ee}}, m, d_{\mathsf{ek}})$ for $(\mathsf{ct}, c^{\mathsf{ee},j}, c_{\mathsf{ek}})$ of $\mathsf{CtMsgRel}^{\mathsf{sk}}$ is given by:

- For $\mathsf{MCFE}^{\mathsf{cpa}}$: for each $i \in [n]$ we need

$$wit^{\mathsf{cpa}}(\widetilde{\mathbf{t}}, i) \overset{def}{=} \Big(\widetilde{\sigma}_{i,0}, \quad -n \cdot \widetilde{\sigma}_{i,0}, \quad \nu \Big) \in \mathbb{Z}_q^3 \tag{14}$$

$$wit^{\mathsf{cpa}}(\mathbf{c}, i) \overset{def}{=} \Big(\sigma_{i,0}, \quad -n \cdot \sigma_{i,0}, \quad \psi_i \Big) \in \mathbb{Z}_q^3 \tag{15}$$

$$wit^{\mathsf{cpa}}(\mathbf{t}, i, k) \overset{def}{=} \Big((\mathbf{s}_i[k] \cdot B_i^{(k)}, \quad \mathbf{u}_i[k] \cdot B_i^{(k)}, \quad \mathbf{x}_i[k])_{k \in [N]}, \quad \nu_i, \rho_i \Big) \in \mathbb{Z}_q^{(2(2N+4)+1)N+2} \tag{16}$$

$$wit^{\mathsf{cpa}}(\mathbf{c}, i, \mathsf{ipfe}) \overset{def}{=} \Big(p_i H_i^{(1)}, \quad p_i H_i^{(2)} \quad \psi_i, \quad (\theta_{i,k})_{k \in [N]} \Big) \in \mathbb{Z}_q^{2(2N+4)+N+1} \tag{17}$$

with respect to the public group elements

$$\Big(\mathbf{g}_1, \quad \mathbf{g}_2, \quad \mathbf{g}_3 \Big) \in \mathbb{G}_1^{3(2N+4)}$$

$$\Big(\mathbf{f}_1, \quad \mathbf{f}_2, \quad \mathbf{f}_3 \Big) \in \mathbb{G}_1^{3(2N+4)}$$

$$\Big([\![\omega]\!]_1, \quad [\![\omega']\!]_1, \quad [\![1]\!]_1, \quad \mathbf{b}_{i,N+1}, \mathbf{b}_{i,N+3} \Big) \in \mathbb{G}_1^{3+2(2N+4)}$$

$$\Big([\![\omega]\!]_1, \quad [\![\omega']\!]_1, \quad \mathbf{h}_{i,3}, \quad \mathbf{h}_{i,N+3+k} \Big) \in \mathbb{G}_1^{2+2(2N+4)} .$$

Over all n clients, the whole witness for EECom is then containing $n(N(4N+9)+5N+15)$ scalars.

- For EECom: let us recall that for each $i \in [n]$, it is defined $M = (M_l)_l = \mathsf{bin}(x_i, z_i) \in \{0,1\}^L$ and $L = \lceil \log x_i \rceil + \lceil \log z_i \rceil$ consists of the binary representation of (x_i, z_i)

$$\forall l \in [L], j \in \{0,1\}: \quad wit^{\mathsf{ee}}(a,i,l,j) \overset{def}{=} (\mathsf{rnd}_{l,j}, M_l^{(i)}) \qquad \in \mathbb{Z}_q \times \{0,1\} \tag{18}$$

$$\forall l \in [L], j \in \{0,1\}: \quad wit^{\mathsf{ee}}(b,i,l,j) \overset{def}{=} s_{l,j}, \mathsf{rnd}_{l,j} \qquad \in \mathbb{Z}_q^2 \tag{19}$$

with respect to the public group elements

$$\begin{array}{lll} g_2, & T & \in \mathbb{G}_2 \times \mathbb{G}_2 \\ (g_1, h_1, W, UV^\sigma), & (0, 0, [\![1]\!]_1, 0) & \in \mathbb{G}_1^4 \times \mathbb{G}_1^4 \ . \end{array}$$

Over all n clients, the whole witness for EECom is then containing $3nL$ scalars.

- For the Groth-Sahai commitment: for each $i \in [n]$ we need

$$\forall j \in [\mathsf{len}(\mathsf{sc}(\mathsf{ek}_i))]: \quad wit^{\mathsf{gs}}(\mathsf{ek}, \mathsf{sc}, i, j) \overset{def}{=} (\mathsf{sc}(\mathsf{ek}_i)[j], s_{i,j}) \qquad \in \mathbb{Z}_q \times \mathbb{Z}_q \tag{20}$$

$$\forall j \in [\mathsf{len}(\mathsf{sc}(\mathsf{ek}_i))]: \quad wit^{\mathsf{gs}}(\mathsf{ek}, \mathsf{grp}, i, j) \overset{def}{=} s_{i,j,1}, s_{i,j,2} \qquad \in \mathbb{Z}_q \times \mathbb{Z}_q \tag{21}$$

which is actually the openings of the commitments

$$c_{\mathsf{ek},i} = \begin{cases} \forall\, j \in [\mathsf{len}(\mathsf{sc}(\mathsf{ek}_i))] : \mathbf{c}_{\mathsf{ek},\mathsf{sc},i,j} & \overset{def}{=} \iota_1'(\mathsf{sc}(\mathsf{ek}_i)[j]) \odot s_{i,j}\mathbf{u}_1 \in \mathbb{G}_1^2, s_{i,j} \overset{\$}{\leftarrow} \mathbb{Z}_q^* \\ \forall\, j \in [\mathsf{len}(\mathsf{sc}(\mathsf{ek}_i))] : \mathbf{d}_{\mathsf{ek},\mathsf{sc},i,j} & \overset{def}{=} \iota_2'(\mathsf{sc}(\mathsf{ek}_i)[j]) \odot s_{i,j}\mathbf{v}_1 \in \mathbb{G}_2^2, s_{i,j} \overset{\$}{\leftarrow} \mathbb{Z}_q^* \\ \forall\, j \in [\mathsf{len}(\mathsf{grp}(\mathsf{ek}_i))] : \mathbf{c}_{\mathsf{ek},\mathsf{grp},i,j} & \overset{def}{=} \iota_1(\mathsf{grp}(\mathsf{ek}_i)[j]) \odot s_{i,j,1}\mathbf{u}_1 \odot s_{i,j,2}\mathbf{u}_2 \in \mathbb{G}_1^2, s_{i,j,1}, s_{i,j,2} \overset{\$}{\leftarrow} \mathbb{Z}_q^* \end{cases}$$

with respect to the public group elements

$$\begin{array}{ll} (u_{2,1}, u_{2,2} + [\![1]\!]_1) \odot \mathbf{u}_1 & \in \mathbb{G}_1 \times \mathbb{G}_1 \\ (v_{2,1}, v_{2,2} + [\![1]\!]_2) \odot \mathbf{v}_1 & \in \mathbb{G}_2 \times \mathbb{G}_2 \\ (1, X) \odot \mathbf{u}_1 \odot \mathbf{u}_2 & \in \mathbb{G}_1 \times \mathbb{G}_1 \end{array}$$

Over all n clients, the whole witness for the Groth-Sahai Com is then containing $n \cdot (2N + (N+2)(N+4) + 2(2N+4))$ scalars for $wit^{\mathsf{gs}}(\mathsf{ek}, \mathsf{sc}, i, j)$ over all $i \in [n]$, and $n \cdot ((N+4) + 6 \cdot (2N+6) + (N+1)(2N+4))$ scalars for $wit^{\mathsf{gs}}(\mathsf{ek}, \mathsf{grp}, i, j)$ over all $i \in [n]$.

D.1 The Σ-protocol for $\mathsf{CtMsgRel}^{\mathsf{sk}}$

Lemma 49. *The Σ-protocol protocol $\Sigma_{\mathsf{cpa},\mathsf{ee}}$ in Figure 8 is* correct, HVZK, special sound *with* high min-entropy, *for proving witness $(wit^{\mathsf{cpa}} \overset{def}{=} (\mathsf{ek}, r), wit^{\mathsf{ee}} \overset{def}{=} (d^{\mathsf{ee}}, m))$ of statement $(\mathsf{ct}_i, c_i^{\mathsf{ee},j})$ with respect to Equation (13) in the relation $\mathsf{CtMsgRel}^{\mathsf{sk}}$.*

Proof. *Correctness* can be inspected from the checks in Figure 9. For showing *high min-entropy*, we consider the first flow from prover

$$\boldsymbol{\Omega}_{wit^{\mathsf{cpa}}(\widetilde{\mathbf{t}},i)}, \boldsymbol{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{c},i)}, \boldsymbol{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})}, \boldsymbol{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{t},i,k)}, \boldsymbol{\Omega}_{wit^{\mathsf{ee}}(a,i,l,j)}, \boldsymbol{\Omega}_{wit^{\mathsf{ee}}(b,i,l,j)} \ .$$

In particular, it suffices to notice that $\mu_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})}$ is sampled uniformly at random in $\mathbb{Z}_q^4$, meanwhile the vectors $[[\![\omega]\!]_1 \,|\, [\![\omega']\!]_1]^\top$ are sampled uniformly at random[23]. Thus the term $\mu_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})}[1,2] \cdot [[\![\omega]\!]_1 \,|\, [\![\omega']\!]_1]^\top$ in

$$\begin{aligned} \boldsymbol{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})} \leftarrow & [\mu_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})}[1,2]] \cdot [[\![\omega]\!]_1 \,|\, [\![\omega']\!]_1]^\top \\ & + [\mu_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})}[3,4]] \cdot [\mathbf{h}_{i,3} | \mathbf{h}_{i,N+3+k}]^\top \end{aligned}$$

is uniformly distributed in $\mathbb{G}_1^2$ and this implies the probability that the adversary successfully guesses $\boldsymbol{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})}$ alone is at most $1/|\mathbb{G}_1|^2 < 2^{-\lambda}$. In the end, this concludes that the

[23] When running on the $\mathsf{MCFE}^{\mathsf{cpa}}$, these $[\![(\omega, \omega')]\!]_1$ are obtained from the random oracle $\mathsf{H}(\mathsf{tag})$.

```
$\mathbf{Prover}(statement = (\mathsf{ct}_i, c_i^{\mathsf{ee},j}); witness = (wit_i^{\mathsf{cpa}}, wit_i^{\mathsf{ee}}))$        $\mathbf{Verifier}(statement)$

1 :   $(\text{For } wit^{\mathsf{cpa}})$
2 :   $\mu_{wit^{\mathsf{cpa}}(\widetilde{\mathbf{t}},i)} \leftarrow \mathbb{Z}_q^3, \mu_{wit^{\mathsf{cpa}}(\mathbf{c},i)} \leftarrow \mathbb{Z}_q^3, \mu_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})} \leftarrow \mathbb{Z}_q^4$
3 :   $\mathbf{\Omega}_{wit^{\mathsf{cpa}}(\widetilde{\mathbf{t}},i)} \leftarrow \mu_{wit^{\mathsf{cpa}}(\widetilde{\mathbf{t}},i)} \cdot \left[\mathbf{g}_1|\mathbf{g}_2|\mathbf{g}_3\right]^\top, \mathbf{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{c},i)} \leftarrow \mu_{wit^{\mathsf{cpa}}(\mathbf{c},i)} \cdot \left[\mathbf{f}_1|\mathbf{f}_2|\mathbf{f}_3\right]^\top$
4 :   $\mathbf{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})} \leftarrow [\mu_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})}[1,2]] \cdot \left[\llbracket\omega\rrbracket_1 \mid \llbracket\omega'\rrbracket_1\right]^\top$
5 :                  $+[\mu_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})}[3,4]] \cdot \left[\mathbf{h}_{i,3}|\mathbf{h}_{i,N+3+k}\right]^\top$
6 :   for $k \in [N]$ do
7 :       $\mu_{wit^{\mathsf{cpa}}(\mathbf{t},i,k)} \leftarrow \mathbb{Z}_q^5, \mathbf{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{t},i,k)} \leftarrow [\mu_{wit^{\mathsf{cpa}}(\mathbf{t},i,k)}[1,2,3]] \cdot \left[\llbracket\omega\rrbracket_1 \mid \llbracket\omega'\rrbracket_1 \mid \llbracket 1\rrbracket_1\right]^\top$
8 :                              $+[\mu_{wit^{\mathsf{cpa}}(\mathbf{t},i,k)}[4,5]] \cdot \left[\mathbf{b}_{i,N+1}|\mathbf{b}_{i,N+3}\right]^\top$
9 :   $(\text{For } wit^{\mathsf{ee}})$
10 :  for $l \in [\lceil\log x_i\rceil + \lceil\log z_i\rceil]$ do
11 :      $\mu_{wit^{\mathsf{ee}}(a,i,l,j)} \leftarrow \mathbb{Z}_q^2, \mathbf{\Omega}_{wit^{\mathsf{ee}}(a,i,l,j)} \leftarrow \mu_{wit^{\mathsf{ee}}(a,i,l,j)} \cdot \left[\llbracket 1\rrbracket_2 \,|T\right]^\top$
12 :      $\mu_{wit^{\mathsf{ee}}(b,i,l,j)} \leftarrow \mathbb{Z}_q^2, \mathbf{\Omega}_{wit^{\mathsf{ee}}(b,i,l,j)} \leftarrow \mu_{wit^{\mathsf{ee}}(b,i,l,j)}[1] \cdot (g_1, h_1, W, U + \sigma \cdot V) + \mu_{wit^{\mathsf{ee}}(b,i,l,j)}[2] \cdot (0, 0, \llbracket 1\rrbracket_1, 0)$

                                $\xrightarrow{\mathbf{\Omega}_{wit^{\mathsf{cpa}}(\widetilde{\mathbf{t}},i)}, \mathbf{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{c},i)}, \mathbf{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})}, \mathbf{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{t},i,k)}}$

                                $\xrightarrow{\mathbf{\Omega}_{wit^{\mathsf{ee}}(a,i,l,j)}, \mathbf{\Omega}_{wit^{\mathsf{ee}}(b,i,l,j)}}$

                                13 :   $\gamma \overset{\$}{\leftarrow} \mathbb{Z}_q$

                                $\xleftarrow{\gamma}$

14 :  $(\text{For } wit^{\mathsf{cpa}})$
15 :  $\tau_{wit^{\mathsf{cpa}}(\widetilde{\mathbf{t}},i)} \leftarrow \mu_{wit^{\mathsf{cpa}}(\widetilde{\mathbf{t}},i)} + \gamma \cdot wit^{\mathsf{cpa}}(\widetilde{\mathbf{t}}, i), \tau_{wit^{\mathsf{cpa}}(\mathbf{c},i)} \leftarrow \mu_{wit^{\mathsf{cpa}}(\mathbf{c},i)} + \gamma \cdot wit^{\mathsf{cpa}}(\mathbf{c}, i)$
16 :  $\tau_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})} \leftarrow \mu_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})} + \gamma \cdot wit^{\mathsf{cpa}}(\mathbf{c}, i, \mathsf{ipfe})$
17 :  for $k \in [N]$ do
18 :      $\tau_{wit^{\mathsf{cpa}}(\mathbf{t},i,k)} \leftarrow \mu_{wit^{\mathsf{cpa}}(\mathbf{t},i,k)} + \gamma \cdot wit^{\mathsf{cpa}}(\mathbf{t}, i, k)$
19 :  $(\text{For } wit^{\mathsf{ee}})$
20 :  for $l \in [\lceil\log x_i\rceil + \lceil\log z_i\rceil]$ do
21 :      $\tau_{wit^{\mathsf{ee}}(a,i,l,j)} \leftarrow \mu_{wit^{\mathsf{ee}}(a,i,l,j)} + \gamma \cdot wit^{\mathsf{ee}}(a, i, l, j)$
22 :      $\tau_{wit^{\mathsf{ee}}(b,i,l,j)} \leftarrow \mu_{wit^{\mathsf{ee}}(b,i,l,j)} + \gamma \cdot wit^{\mathsf{ee}}(b, i, l, j)$

                                $\xrightarrow{\tau_{wit^{\mathsf{cpa}}(\widetilde{\mathbf{t}},i)}, \tau_{wit^{\mathsf{cpa}}(\mathbf{c},i)}, \tau_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})}, \tau_{wit^{\mathsf{cpa}}(\mathbf{t},i,k)}}$

                                $\xrightarrow{\tau_{wit^{\mathsf{ee}}(a,i,l,j)}, \tau_{wit^{\mathsf{ee}}(b,i,l,j)}}$

                                23 :   Perform the checks in Figure 9
```

Fig. 8: Description of $\Sigma_{\mathsf{cpa},\mathsf{ee}}$ for proving the parts of $\mathsf{CtMsgRel}^{\mathsf{sk}}$ that involves wit^{cpa} and wit^{ee}, regarding Equation (13) for $(\mathsf{ct}_i, c_i^{\mathsf{ee},j})$. The checks by the **Verifier** are detailed in Figure 9.

Input: The vectors $\tau_{wit^{\mathsf{cpa}}(\widetilde{\mathbf{t}},i)}, \tau_{wit^{\mathsf{cpa}}(\mathbf{c},i)}, \tau_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})}, \tau_{wit^{\mathsf{cpa}}(\mathbf{t},i,k)}$, $\tau_{wit^{\mathsf{ee}}(a,i,l,j)}, \tau_{wit^{\mathsf{ee}}(b,i,l,j)}$ and $\mathbf{\Omega}_{wit^{\mathsf{cpa}}(\widetilde{\mathbf{t}},i)}, \mathbf{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{c},i)}, \mathbf{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})}, \mathbf{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{t},i,k)}$, and $\mathbf{\Omega}_{wit^{\mathsf{ee}}(a,i,l,j)}, \mathbf{\Omega}_{wit^{\mathsf{ee}}(b,i,l,j)}$ that are received from the **Prover**.

1. (For wit_i^{cpa}) Verify the following:

$$\begin{aligned}
\mathbf{\Omega}_{wit^{\mathsf{cpa}}(\widetilde{\mathbf{t}},i)} + \gamma \cdot \widetilde{\mathbf{t}}_{i,0} &\stackrel{?}{=} \tau_{wit^{\mathsf{cpa}}(\widetilde{\mathbf{t}},i)} \cdot [\mathbf{g}_1|\mathbf{g}_2|\mathbf{g}_3]^\top \\
\mathbf{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{c},i)} + \gamma \cdot \mathbf{c}_{i,0} &\stackrel{?}{=} \tau_{wit^{\mathsf{cpa}}(\mathbf{c},i)} \cdot [\mathbf{f}_1|\mathbf{f}_2|\mathbf{f}_3]^\top \\
\mathbf{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})} + \gamma \cdot \mathbf{c}_{i,\mathsf{ipfe}} &\stackrel{?}{=} \tau_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})}[1,2] \cdot [\llbracket\omega\rrbracket_1 \mid \llbracket\omega'\rrbracket_1]^\top \\
&\quad + \tau_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})}[3,4] \cdot [\mathbf{h}_{i,3}|\mathbf{h}_{i,N+3+k}]^\top \\
\mathbf{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{t},i,k)} + \gamma \cdot \mathbf{t}_i &\stackrel{?}{=} \tau_{wit^{\mathsf{cpa}}(\mathbf{t},i,k)}[1,2,3] \cdot [\llbracket\omega\rrbracket_1 \mid \llbracket\omega'\rrbracket_1 \mid \llbracket 1\rrbracket_1]^\top \\
&\quad + \tau_{wit^{\mathsf{cpa}}(\mathbf{t},i,k)}[4,5] \cdot [\mathbf{b}_{i,N+1}|\mathbf{b}_{i,N+3}]^\top
\end{aligned}$$

2. (For wit_i^{ee}) Verify the following

$$\begin{aligned}
\mathbf{\Omega}_{wit^{\mathsf{ee}}(a,i,l,j)} + \gamma \cdot \mathbf{a}_l^{(i)} &\stackrel{?}{=} \tau_{wit^{\mathsf{ee}}(a,i,l,j)} \cdot [\llbracket 1\rrbracket_2 \, |T]^\top \\
\mathbf{\Omega}_{wit^{\mathsf{ee}}(b,i,l,j)} + \gamma \cdot (\mathbf{b}_{l,j}^{(i)}) &\stackrel{?}{=} \tau_{wit^{\mathsf{ee}}(b,i,l,j)} \cdot (g_1, h_1, W, U + \sigma V) + \tau_{wit^{\mathsf{ee}}(b,i,l,j)} \cdot (0, 0, \llbracket 1\rrbracket_1, 0)
\end{aligned}$$

Fig. 9: Description of $\Sigma_{\mathsf{cpa,ee}}$ - checks by **Verifier**.

probability of guessing correctly the whole commitment is at most $2^{-\lambda}$ and concludes the proof for *high min-entropy*.

For *HVZK*, we observe that the openings are computed over $\mathbb{Z}_q$ and as all commitments' masks μ are sampled uniformly at random with elements in $\mathbb{Z}_q$, the third flow (that comes from computations between lines 15-22) from the prover is distributed uniformly at random as vectors over $\mathbb{Z}_q$. Therefore, from the checks in Figure 9, upon the challenge γ and the openings, the first flow is uniquely determined

$$\begin{aligned}
\mathbf{\Omega}_{wit^{\mathsf{cpa}}(\widetilde{\mathbf{t}},i)} &= \tau_{wit^{\mathsf{cpa}}(\widetilde{\mathbf{t}},i)} \cdot [\mathbf{g}_1|\mathbf{g}_2|\mathbf{g}_3]^\top - \gamma \cdot \widetilde{\mathbf{t}}_{i,0} \\
\mathbf{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{c},i)} &= \tau_{wit^{\mathsf{cpa}}(\mathbf{c},i)} \cdot [\mathbf{f}_1|\mathbf{f}_2|\mathbf{f}_3]^\top - \gamma \cdot \mathbf{c}_{i,0} \\
\mathbf{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})} &= \tau_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})}[1,2] \cdot [\llbracket\omega\rrbracket_1 \mid \llbracket\omega'\rrbracket_1]^\top \\
&\quad + \tau_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})}[3,4] \cdot [\mathbf{h}_{i,3}|\mathbf{h}_{i,N+3+k}]^\top - \gamma \cdot \mathbf{c}_{i,\mathsf{ipfe}} \\
\mathbf{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{t},i,k)} &= \tau_{wit^{\mathsf{cpa}}(\mathbf{t},i,k)}[1,2,3] \cdot [\llbracket\omega\rrbracket_1 \mid \llbracket\omega'\rrbracket_1 \mid \llbracket 1\rrbracket_1]^\top \\
&\quad + \tau_{wit^{\mathsf{cpa}}(\mathbf{t},i,k)}[4,5] \cdot [\mathbf{b}_{i,N+1}|\mathbf{b}_{i,N+3}]^\top - \gamma \cdot \mathbf{t}_i \\
\mathbf{\Omega}_{wit^{\mathsf{ee}}(a,i,l,j)} &= \tau_{wit^{\mathsf{ee}}(a,i,l,j)} \cdot [\llbracket 1\rrbracket_2 \, |T]^\top - \gamma \cdot \mathbf{a}_l^{(i)} \\
\mathbf{\Omega}_{wit^{\mathsf{ee}}(b,i,l,j)} &= \tau_{wit^{\mathsf{ee}}(b,i,l,j)} \cdot (g_1, h_1, W, U\sigma \cdot V) + \tau_{wit^{\mathsf{ee}}(b,i,l,j)} \cdot (0, 0, \llbracket 1\rrbracket_1, 0) \\
&\quad - \gamma \cdot (\mathbf{b}_{l,j}^{(i)}) \ .
\end{aligned}$$

Give the above discussion, the HVZK simulator, when given the challenge γ, samples the openings and compute the commitments as above. The produced transcript is then indistinguishable from the real one.

Finally, let us show the 2 *special soundness* property. We want to construct a deterministic PT extractor that can extract the witness $\omega_i^{\mathsf{cpa}}, \omega_i^{\mathsf{ee}}$ given 2 valid transcripts that are indexed by $e \in [2]$

Identical first flow:

$\mathbf{\Omega}_{wit^{\mathsf{cpa}}(\widetilde{\mathbf{t}},i),e}, \mathbf{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{c},i),e}, \mathbf{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe}),e}, \mathbf{\Omega}_{wit^{\mathsf{cpa}}(\mathbf{t},i,k),e},$

$\mathbf{\Omega}_{wit^{\mathsf{ee}}(a,i,l,j),e}, \mathbf{\Omega}_{wit^{\mathsf{ee}}(b,i,l,j),e}$

Pairwise distinct challenge: γ_e

Third flow:

$\tau_{wit^{\mathsf{cpa}}(\widetilde{\mathbf{t}},i),e}, \tau_{wit^{\mathsf{cpa}}(\mathbf{c},i),e}, \tau_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe}),e}, \tau_{wit^{\mathsf{cpa}}(\mathbf{t},i,k),e}, \tau_{wit^{\mathsf{ee}}(a,i,l,j),e}, \tau_{wit^{\mathsf{ee}}(b,i,l,j),e}$

the extractor extracts the witness

$$witness \overset{def}{=} \left(wit^{\mathsf{cpa}}(\widetilde{\mathbf{t}}, i), wit^{\mathsf{cpa}}(\mathbf{c}, i), wit^{\mathsf{cpa}}(\mathbf{t}, i, k), wit^{\mathsf{ee}}(a, i, l, j), wit^{\mathsf{ee}}(b, i, l, j)\right) \ .$$

In the following, we denote by $\Delta v_{e,f} \overset{def}{=} \tau_{v,e} - \tau_{v,f} \in \mathbb{Z}$ for $v \in witness$ and $\Delta\gamma_{e,f} \overset{def}{=} \gamma_e - \gamma_f \neq 0$. The extraction is done *a la Schnorr*, by remarking that the inverses of $\Delta\gamma_{e,f}$ exist in $\mathbb{Z}_q$. The main equations to resolve for witnesses are the following:

$$\begin{aligned}
\Delta\gamma_{1,2} \cdot \widetilde{\mathbf{t}}_{i,0} &= \Delta\tau_{wit^{\mathsf{cpa}}(\widetilde{\mathbf{t}},i)}{}_{1,2} \cdot \left[\mathbf{g}_1 | \mathbf{g}_2 | \mathbf{g}_3\right]^\top \\
\Delta\gamma_{1,2} \cdot \mathbf{c}_{i,0} &= \Delta\tau_{wit^{\mathsf{cpa}}(\mathbf{c},i)}{}_{1,2} \cdot \left[\mathbf{f}_1 | \mathbf{f}_2 | \mathbf{f}_3\right]^\top \\
\Delta\gamma_{1,2} \cdot \mathbf{c}_{i,\mathsf{ipfe}} &= \Delta\tau_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})}[1,2]_{1,2} \cdot \left[[\![\omega]\!]_1 \,|\, [\![\omega']\!]_1\right]^\top \\
&\quad + \Delta\tau_{wit^{\mathsf{cpa}}(\mathbf{c},i,\mathsf{ipfe})}[3,4]_{1,2} \cdot \left[\mathbf{h}_{i,3} | \mathbf{h}_{i,N+3+k}\right]^\top \\
\Delta\gamma_{1,2} \cdot \mathbf{t}_i &= \Delta\tau_{wit^{\mathsf{cpa}}(\mathbf{t},i,k)}[1,2,3]_{1,2} \cdot \left[[\![\omega]\!]_1 \,|\, [\![\omega']\!]_1 \,|\, [\![1]\!]_1\right]^\top \\
&\quad + \Delta\tau_{wit^{\mathsf{cpa}}(\mathbf{t},i,k)}[4,5]_{1,2} \cdot \left[\mathbf{b}_{i,N+1} | \mathbf{b}_{i,N+3}\right]^\top \\
\Delta\gamma_{1,2} \cdot \mathbf{a}_l^{(i)} &= \Delta\tau_{wit^{\mathsf{ee}}(a,i,l,j)}{}_{1,2} \cdot \left[[\![1]\!]_2 \,| T\right]^\top \\
\Delta\gamma_{1,2} \cdot (\mathbf{b}_{l,j}^{(i)}) &= \Delta\tau_{wit^{\mathsf{ee}}(b,i,l,j)}[1]_{1,2} \cdot (g_1, h_1, W, U + \sigma \cdot V) + \Delta\tau_{wit^{\mathsf{ee}}(b,i,l,j)}[2]_{1,2}(0, 0, [\![1]\!]_1, 0) \ .
\end{aligned}$$

Canceling $\Delta\gamma_{1,2}$ in the above equations, we can solve for the witnesses. $\square$

E Recall on Groth-Sahai NIZK for Section 4.2

Recall: Three types of equations in Groth-Sahai. We have three types of equations that can be proved using the Groth-Sahai NIZK:

1. A *pairing-product equation* over variables $\vec{\mathcal{X}} = (\mathcal{X}_1, \ldots, \mathcal{X}_m) \in \mathbb{G}_1^m$ and $\vec{\mathcal{Y}} = (\mathcal{Y}_1, \ldots, \mathcal{Y}_n) \in \mathbb{G}_2^n$:
$$\langle \vec{\mathcal{A}}, \vec{\mathcal{Y}} \rangle + \langle \vec{\mathcal{X}}, \vec{\mathcal{B}} \rangle + \langle \vec{\mathcal{X}}, \Gamma \cdot \vec{\mathcal{Y}} \rangle = t_{\mathrm{t}} \ , \tag{22}$$
where $\vec{\mathcal{A}} \in \mathbb{G}_1^n$, $\vec{\mathcal{B}} \in \mathbb{G}_2^m$, and $\Gamma \in \mathbb{Z}_q^{m \times n}$, and $t_{\mathrm{t}} \in \mathbb{G}_{\mathrm{t}}$.
2. A *multi-scalar equation* over variables $\vec{\mathcal{X}} = (\mathcal{X}_1, \ldots, \mathcal{X}_m) \in \mathbb{G}_1^m$ and $\mathbf{y} \in \mathbb{Z}_q^n$:
$$\langle \vec{\mathcal{A}}, \vec{\mathbf{y}} \rangle + \langle \vec{\mathcal{X}}, \mathbf{b} \rangle + \langle \vec{\mathcal{X}}, \Gamma \cdot \mathbf{y} \rangle = T \ , \tag{23}$$
where $\vec{\mathcal{A}} \in \mathbb{G}_1^n$, $\mathbf{b} \in \mathbb{Z}_q^m$, $\Gamma \in \mathbb{Z}_q^{m \times n}$, and $T \in \mathbb{G}_1$. Similar multi-scalar equations can be defined for $\mathbb{G}_2$.
3. A *quadratic equation* over variables $\mathbf{x} \in \mathbb{Z}_q^m$ and $\mathbf{y} \in \mathbb{Z}_q^n$:
$$\langle \mathbf{a}, \mathbf{y} \rangle + \langle \mathbf{x}, \mathbf{b} \rangle + \langle \mathbf{x}, \Gamma \cdot \mathbf{y} \rangle = t \ , \tag{24}$$
where $\mathbf{a} \in \mathbb{Z}_q^n$, $\mathbf{b} \in \mathbb{Z}_q^m$, $\Gamma \in \mathbb{Z}_q^{m \times n}$, and $t \in \mathbb{Z}_q$.

Recall: The Groth-Sahai NIZK from SXDH *and its Commitment.* For the sake of verifying correctness of the encryption keys, which lie in $\mathbb{G}_1$, we focus on the multi-scalar equations in (23) for Groth-Sahai. The setup algorithm GenCRS generates the public parameters $\mathsf{pp}^{\mathsf{gs}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_{\mathrm{t}}, g_1, g_2, g_{\mathrm{t}}, \mathbf{e}, q)$ and a commitment key that contains[24]:

$$\begin{aligned}
\mathbf{u}_1 &:= (u_{1,1}, u_{1,2}) \in \mathbb{G}_1^2, & \mathbf{u}_2 &:= (u_{2,1}, u_{2,2}) \in \mathbb{G}_1^2 \\
\mathbf{v}_1 &:= (v_{1,1}, v_{1,2}) \in \mathbb{G}_1^2, & \mathbf{v}_2 &:= (v_{2,1}, v_{2,2}) \in \mathbb{G}_2^2
\end{aligned}$$

[24] In the following we write the bilinear group setting additively to be coherent with the notations of the MCFE in Fig. 6.

We write $\overrightarrow{\mathcal{U}} = (\mathbf{u}_1, \mathbf{u}_2)^\top$ and $\overrightarrow{\mathcal{V}} = (\mathbf{v}_1, \mathbf{v}_2)^\top$ and also define the following maps:

$$\begin{aligned}&\text{Let } X \in \mathbb{G}_1, Y \in \mathbb{G}_2 :\\ &\quad \iota_1(X) = (1, X), \quad \iota_2(Y) = (1, Y), \iota_1'(x) = (u_{2,1}^x, (u_{2,2} + [\![1]\!]_1)^x), \iota_2'(x) = (v_{2,1}^x, (v_{2,2} + [\![1]\!]_2)^x) \ .\end{aligned} \quad (25)$$

The commitment key is $\mathsf{pp}^{\mathsf{com}} = (\overrightarrow{\mathcal{U}}, \overrightarrow{\mathcal{V}}, \iota_1(\cdot), \iota_2(\cdot), \iota_1'(\cdot), \iota_2'(\cdot))$. Given $\mathsf{pp}^{\mathsf{com}}$, the commitment algorithm $\mathsf{Com.Commit}$ works differently on different type of committed values for Groth-Sahai: the operator $\odot$ denotes the Hadamard product,

1. For a type $\mathsf{grp\text{-}one}$ and $X \in \mathbb{G}_1$, it sets the opening $d = (s_1, s_2, X)$ where $s_1, s_2 \xleftarrow{\$} \mathbb{Z}_q^*$ and the commitment is $\mathbf{c}_X = \iota_1(X) \odot s_1\mathbf{u}_1 \odot s_2\mathbf{u}_2 \in \mathbb{G}_1^2$.
2. For a type $\mathsf{grp\text{-}two}$ and $Y \in \mathbb{G}_2$, it sets the opening $d = (s_1, s_2, Y)$ where $s_1, s_2 \xleftarrow{\$} \mathbb{Z}_q^*$ and the commitment is $\mathbf{c}_Y = \iota_2(Y) \odot s_1\mathbf{v}_1 \odot s_2\mathbf{v}_2$.
3. For a type scalar and $x \in \mathbb{Z}_q$, it sets the opening $d = (s, x)$ where $s \xleftarrow{\$} \mathbb{Z}_q^*$, and the commitment contains two elements: $\mathbf{c}_x = \iota_1'(x) \odot s\mathbf{u}_1 \in \mathbb{G}_1^2$ and $\mathbf{d}_x = \iota_2'(x) \odot s\mathbf{v}_1 \in \mathbb{G}_2^2$.

E.1 Batch Verification of Groth-Sahai NIZK

To detail the verification of the Groth-Sahai NIZK, following the notations in [20], we specify some relations: let $t \in \mathbb{Z}_q, T_1 \in \mathbb{G}_1, T_2 \in \mathbb{G}_2$, and $t_{\mathsf{t}} \in \mathbb{G}_{\mathsf{t}}$

$$\iota_{\mathsf{t}}(t_{\mathsf{t}}) := \begin{bmatrix} 1 & 1 \\ 1 & t_{\mathsf{t}} \end{bmatrix}, \ \hat{\iota}(T_1)_{\mathsf{t}} := \begin{bmatrix} 1 & 1 \\ e(T_1, v_{2,1}) & e(T_1, v_{2,2} + [\![1]\!]_2) \end{bmatrix}, \ \hat{\iota}(T_2)_{\mathsf{t}} := \begin{bmatrix} 1 & e(u_{2,1}, T_2) \\ 1 & e(u_{2,2} + [\![1]\!]_1, T_2) \end{bmatrix}$$

and

$$\iota_{\mathsf{t}}(t) := \begin{bmatrix} e(u_{2,1}, v_{2,1}) \cdot t & e(u_{2,1}, v_{2,2} + [\![1]\!]_2) \cdot t \\ e(u_{2,2} + [\![1]\!]_1, v_{2,1}) \cdot t & e(u_{2,2} + [\![1]\!]_1, v_{2,2} + [\![1]\!]_2) \cdot t \end{bmatrix} .$$

For the sake of notation, for $\mathbf{c} = (c_1, c_2) \in \mathbb{G}_1^2$ and $\mathbf{d} = (d_1, d_2) \in \mathbb{G}_2^2$, we define $\mathbf{c} \bullet \mathbf{d} \overset{def}{=} \begin{bmatrix} e(c_1, d_1) & e(c_1, d_2) \\ e(c_2, d_1) & e(c_2, d_2) \end{bmatrix}$ and similarly to [49], we denote $F(\mathbf{c}, \mathbf{d}) \overset{def}{=} \mathbf{c} \bullet \mathbf{d}$.

We consider in $\Pi^{\mathsf{ss}}_{\mathsf{gs,cpa,ee}}$ only multi-scalar multiplication in $\mathbb{G}_1$. The commitment key is $\mathsf{pp}^{\mathsf{com}} = (\overrightarrow{\mathcal{U}}, \overrightarrow{\mathcal{V}}, \iota_1(\cdot), \iota_2(\cdot), \iota_1'(\cdot), \iota_2'(\cdot))$. As a reminder, a *multi-scalar equation* over variables $\overrightarrow{\mathcal{X}} = (\mathcal{X}_1, \ldots, \mathcal{X}_m) \in \mathbb{G}_1^m$ and $\mathbf{y} \in \mathbb{Z}_q^n$:

$$\langle \overrightarrow{\mathcal{A}}, \overrightarrow{\mathbf{y}} \rangle + \langle \overrightarrow{\mathcal{X}}, \mathbf{b} \rangle + \langle \overrightarrow{\mathcal{X}}, \Gamma \cdot \mathbf{y} \rangle = T \ , \quad (26)$$

where $\overrightarrow{\mathcal{A}} \in \mathbb{G}_1^n$, $\mathbf{b} \in \mathbb{Z}_q^m$, $\Gamma \in \mathbb{Z}_q^{m \times n}$, and $T \in \mathbb{G}_1$. The verification of a proof

$$(\mathbf{c}^{\mathsf{gs}}, \mathbf{d}^{\mathsf{gs}}, \mathbf{pi}^{\mathsf{gs}}, \theta^{\mathsf{gs}}) \in \mathbb{G}_1^{m \times 2} \times \mathbb{G}_2^{n \times 2} \times \mathbb{G}_2^{2 \times 2} \times \mathbb{G}_1^{1 \times 2}$$

necessitates to check the following: by denoting $\odot$ the Hadamard product, and using $\bullet$ that is defined as above,

$$\left[\iota_1(\overrightarrow{\mathcal{A}}) \bullet \mathbf{d}^{\mathsf{gs}}\right] \odot \left[\mathbf{c}^{\mathsf{gs}} \bullet \iota_2'(\mathbf{b})\right] \odot \left[\mathbf{c}^{\mathsf{gs}} \bullet \Gamma \cdot \mathbf{d}^{\mathsf{gs}}\right] \overset{?}{=} \hat{\iota}_{\mathsf{t}}(T_1) \odot \left[\overrightarrow{\mathcal{U}} \bullet \mathbf{pi}^{\mathsf{gs}}\right] \odot F(\theta^{\mathsf{gs}}, \mathbf{v}_1) \ ,$$

where $\mathbf{v}_1 = (v_{2,1}, v_{2,2})$ comes from $\overrightarrow{\mathcal{V}}$. The verification can be batched, following [20], to obtain $\min(2m + n, 2n + 2)$ pairings to compute the left hand side, depending on how we rewrite the expression. On the right hand side, the same technique leads to 7 pairings.

F Proof of Lemma 32

Lemma 32. *Let $\mathcal{F}$ be a function class with public inputs $(\mathcal{Z}_{\lambda,i})_{i\in[n]}$ where $\mathcal{Z}_{\lambda,i} := \mathsf{Tag} \times \widetilde{\mathcal{Z}}_{\lambda,i}$ for some tag space $\mathsf{Tag} = \{0,1\}^{poly(\lambda)}$. The elements of $\mathcal{F}$ are $F_{\lambda,n} : \prod_{i=1}^{n} (\mathcal{D}_{\lambda,i} \times \mathcal{Z}_{\lambda,i}) \to \mathcal{R}_\lambda$. Suppose that $\mathcal{F}$ contains the identity function $F^{\mathsf{id}}_{\lambda,n}$ where for all $(x_i, z_i)_i$, $F^{\mathsf{id}}_{\lambda,n}((x_i, z_i)_i) = (x_i, z_i)_i$. If there exists a* CCA-secure *MCFE scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$ for the function class $\mathcal{F}$ with public inputs, then there exists a* CCA-secure *FE scheme for the same function class $\mathcal{F}$.*

Proof. Suppose that $\mathsf{MCFE}^{\mathsf{xxx}\text{-}}[\mathcal{F}, (\mathcal{Z}_{\lambda,i})_{i\in[n]}] = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$, where $\mathsf{xxx} \in \{\mathsf{sel}, \mathsf{adp}, \mathsf{stat}\}$, is a *CCA-secure* MCFE scheme for the function class $\mathcal{F}$ with public inputs, following Definition 30.

The function class is $\mathcal{F}$ containing $F_\lambda : \mathcal{D}_\lambda \times \mathcal{Z}_\lambda \to \mathcal{R}_\lambda$. Following Definition 43, the obtained $\mathsf{FE}^{\mathsf{xxx}\text{-}}[\mathcal{F}, (\mathcal{Z}_{\lambda,i})_{i\in[n]}]$ is defined by algorithms:

$\mathsf{Setup}^{\mathsf{pk}}(1^\lambda)$**:** Run $\mathsf{Setup}^{\mathsf{mc}}(1^\lambda, 1^1) \to (\mathsf{msk}^{\mathsf{mc}}, \mathsf{ek}^{\mathsf{mc}})$. Output $\mathsf{msk} := \mathsf{msk}^{\mathsf{mc}}, \mathsf{pk} := (\mathsf{ek}^{\mathsf{mc}})$.
$\mathsf{Extract}^{\mathsf{pk}}(\mathsf{msk}, F_\lambda)$**:** Run $\mathsf{Extract}^{\mathsf{mc}}(\mathsf{msk}^{\mathsf{mc}}, F_\lambda) \to \mathsf{dk}_{F_\lambda}$ and output dk_{F_λ}.
$\mathsf{Enc}^{\mathsf{pk}}(\mathsf{pk}, x, z)$**:** Parse $\mathsf{pk} := (\mathsf{ek}^{\mathsf{mc}})$ and $z := (\epsilon, \tilde{z})$ as there is no tag in single client and public key FE. Sample $\mathsf{tag} \overset{\$}{\leftarrow} \mathsf{Tag}$ and run $\mathsf{Enc}^{\mathsf{mc}}(\mathsf{ek}^{\mathsf{mc}}, x, (\mathsf{tag}, \tilde{z})) \to \mathsf{ct}$. Finally output ct.
$\mathsf{Dec}^{\mathsf{pk}}(\mathsf{dk}_{F_\lambda}, \mathsf{ct})$**:** Run and output $\mathsf{Dec}^{\mathsf{mc}}(\mathsf{dk}_{F_\lambda}, \mathsf{ct})$.

Correctness follows from the correctness of the MCFE. If the function class captures access control, then the FE is for the same class having access control as well. In terms of security, let $\mathcal{A}$ be an adversary against the FE *as per* Definition 44. We construct an adversary $\mathcal{B}$ breaking $\mathsf{MCFE}^{\mathsf{xxx\text{–}rep\text{-}priv}}[\mathcal{F}, (\mathcal{Z}_{\lambda,i})_{i\in[n]}]$, with *static* corruptions, using $\mathcal{A}$. The adversary $\mathcal{B}$ simulates the IND-CCA FE game by (i) first querying its MCFE challenger on $(1^\lambda, 1)$ to obtain the public parameters pp (if any) then *queries* **Corrupt**(1), gets ek, and forwards $\mathsf{pk} := \mathsf{ek}$ together with pp to $\mathcal{A}$. We note that the corrupted client is known from the beginning; (ii) simulating the FE's challenge queries by forwarding the challenge queries (*i.e.* sample $\mathsf{tag} \overset{\$}{\leftarrow} \mathsf{Tag}$ and define the challenge to be $(1, x^{(0)}, x^{(1)}, (\mathsf{tag}, \tilde{z}^{(chal)})))$ to its MCFE challenger given $\left(\left(x^{(0)}, (\epsilon, \tilde{z}^{(chal)})\right), \left(x^{(1)}, (\epsilon, \tilde{z}^{(chal)})\right)\right)$ by $\mathcal{A}$; (iii) the key extraction and decryption queries are forwarded to the MCFE challenger in a straightforward manner, thanks to the fact that each decryption contains a fully specified FE ciphertext corresponding to 1 client in the MCFE.

If the FE adversary $\mathcal{A}$ is admissible, *i.e.* $x^{(0)} \neq x^{(1)}$ but $F(x^{(0)}, (\epsilon, \tilde{z}^{(chal)})) = F(x^{(1)}, (\epsilon, \tilde{z}^{(chal)}))$ for all F queried to **Extract**, then the challenge query

$$(1, x^{(0)}, x^{(1)}, (\mathsf{tag}, \tilde{z}^{(chal)}))$$

is on a pair of inputs $(x^{(0)}, (\mathsf{tag}, \tilde{z}^{(chal)})) \neq (x^{(1)}, (\mathsf{tag}, \tilde{z}^{(chal)}))$ conforming to the admissibility. This implies that $\mathcal{B}$ is also admissible following Definition 29. Moreover, the fact that every encryption query is defined on a *freshly sampled* tag implies that there is no repetitions for any pair $(1, \mathsf{tag})$ registered to the MCFE challenger. This allows us to allow encrypting different public inputs, even when the MCFE is for *private inputs repetitions* only (this is the case of the concrete MCFE constructed later in Section 4.2, since we use [67]). Therefore, if $\mathcal{A}$ wins the FE game, then $\mathcal{B}$ wins the MCFE game. □

G Proof of Theorem 33

Theorem 33. *If the* admissibility *of $\mathcal{F}$ is efficiently decidable and: $\mathsf{MCFE}^{\mathsf{cpa}}[\mathcal{F}, (\mathcal{Z}_{\lambda,i})_{i\in[n]}]$ is* IND-CPA *secure;* Com *is a* perfectly binding *and* computationally hiding *commitment scheme; PRF is a secure PRF;* NIZK *is a NIZK for the relation* $\mathsf{CtMsgRel}^{\mathsf{sk}}$*, satisfying* correctness, zero-knowledge*, and* simulation soundness*;* EECom *is an* equivocable-extractable *commitment scheme; then $\mathsf{MCFE}^{\mathsf{cca}}[\mathcal{F}, (\mathcal{Z}_{\lambda,i})_{i\in[n]}]$ is* pos-statically *IND-CCA secure.*

Proof. We give the main ideas of the game transitions and recall that the adversary $\mathcal{A}$ is *pos*-restricted, with only *one* challenge tag tag^* and under *static* corruptions. The changes that make the transitions between games are highlighted in gray. The advantage of an adversary $\mathcal{A}$ in a game G_i is denoted by

$$\mathsf{Adv}(\mathsf{G}_i) := \Pr[\mathsf{G}_i = 1] \enspace .$$

G_0**:** This is the original security game with a challenge bit $b \xleftarrow{\$} \{0,1\}$, allowing static corruption, with *pos*-restriction, and one challenge tag tag^*. For a fixed (i, tag), the adversary is allowed repetitive queries $(x_i^{(j)}, z_i^{(j)})$, which are indexed by j. We can furthermore suppose that the challenge tag tag^* is queried to the **LoR** oracle and not to **Enc**, this incurs a polynomial multiplicative loss in the advantage of the adversary. This follows a reduction from the full-fledged no-restriction original security game, in which the simulator makes a guess on which tag to **Enc** will be tag^* and the simulator aborts if the guess is not correct. Then, any **Enc** query on tag^* for $(i, x_i, z_i, \mathsf{tag}^*)$ is answered by **LoR** on $(i, (x_i^{(0)}, z_i^{(0)}) \overset{def}{=} (x_i, z_i), (x_i^{(1)}, z_i^{(1)}) \overset{def}{=} (x_i, z_i), \mathsf{tag}^*)$. There are only polynomially many tags, which bounds the probability of abort that is an event independent from the success of breaking the original no-restriction game.

G_1**:** We switch the crs of the commitment to equivocable-extractable mode, use it to simulate the decryption oracle. More specifically, the simulator runs $\mathsf{SetupTr}(1^\lambda)$ to obtain the trapdoor td, for a simulated $\widetilde{\mathsf{crs}}$, so as to set up the public parameters. The trapdoor td is used to extract the message from the commitment $c^{\mathsf{ee},j}_{\mathsf{tag},i}$ in $\mathsf{ct}_i^{(j)} := (\mathsf{ct}^{\mathsf{cpa}}, j_i, c^{\mathsf{ee},j}_{\mathsf{tag},i}, \pi^{(j)}_{\mathsf{tag},i})$, for $i \in \mathcal{C}$ in the decryption queries by the adversary. Additionnally, td also allows us to simulate commitments and later open them to our desired values. In short, these properties of td and the simulation modes are provided by the strong properties: *Strong Simulation Indistinguishability* and *Strong Binding Extractability* from Definition 2 on the E2-security of EECom. Any adversary that distinguishes G_0 from G_1 can thus be used to distinguish the set up algorithms $\mathsf{EECom.Setup}$ and $\mathsf{EECom.SetupTr}$.

G_2**:** We define an event $\mathsf{Fail}_{\mathsf{tag}^*}$ as follows:

There exist $(F, \mathbf{c}) \in \mathcal{Q}_{\mathsf{Dec}}$ such that there exists $i \in \mathcal{C}$ with $\mathsf{ct}_i^{(j)} := (\mathsf{ct}^{\mathsf{cpa},j}_{\mathsf{tag}^*,i}, \mathsf{ct}^{\mathsf{pke}}_{\mathsf{tag}^*,i}, \pi^{(j)}_{\mathsf{tag}^*,i}) \in \mathbf{c}$ while given the challenge bit b,

$$\begin{cases} \mathsf{Verify}^{\mathsf{H}}(\mathsf{crs}, (\mathsf{ct}^{\mathsf{cpa},j}_{\mathsf{tag},i}, c^{\mathsf{ee},j}_{\mathsf{tag},i}, c^{\mathsf{cpa},j}_{\mathsf{ek},i}), \pi^{(j)}_{\mathsf{tag}^*,i}) & = 1 \\ \mathsf{ct}^{\mathsf{cpa},j}_{\mathsf{tag}^*,i} & = \mathsf{Enc}^{\mathsf{cpa}}(\mathsf{ek}_i^{\mathsf{cpa}}, x_i^{(j)}, z_i^{(j)}; r^{(j)}_{\mathsf{tag}^*,i}) \\ z_i^{(j)} & = (\mathsf{tag}^*, \tilde{z}_i^{(j)}) \\ \mathcal{A} \text{ is admissible} & \end{cases}$$

$$\mathsf{AND} \quad \begin{cases} \mathsf{Com.Verify}(\mathsf{pp}^{\mathsf{com}}, c_{\mathsf{ek},i}, \widetilde{\mathsf{ek}_i^{\mathsf{cpa}}}, d_{\mathsf{ek},i}) & = 1 \text{ for } \widetilde{\mathsf{ek}_i^{\mathsf{cpa}}} \neq \mathsf{ek}_i^{\mathsf{cpa}} \\ \mathsf{OR} \quad \mathsf{EECom.ExtCom}(\mathsf{td}, c^{\mathsf{ee},j}_{\mathsf{tag},i}) & \neq x_i^{(j)} \end{cases} ,$$

where the randomness $r^{(j)}_{\mathsf{tag}^*,i} \leftarrow \mathsf{PRF}(K_i, x_i^{(j)}, z_i^{(j)}, \mathsf{tag}^*)$.

The **Finalise** procedure checks in addition if $\mathsf{Fail}_{\mathsf{tag}^*}$ happens, we abort if it is the case. We note that the check $\mathsf{EECom.ExtCom}(\mathsf{td}, c^{\mathsf{ee},j}_{\mathsf{tag},i}) \overset{?}{=} x_i$ can be done thanks to the trapdoor td resulting from the simulated set up for EECom since G_1.

We are now going to aruge that introducing this abort does not significantly change the view of the adversary. First of all, thanks to the *perfect binding property* of the commitment Com, no ppt adversary can produce a valid proof $\pi_{\mathsf{tag}^*,i}$ that

- $\pi_{\mathsf{tag}^*,i}$ verifies on a ciphertext $(\mathsf{ct}^{\mathsf{cpa}}_{\mathsf{tag},i}, c^{\mathsf{ee},j}_{\mathsf{tag},i}, c^{\mathsf{cpa}}_{\mathsf{ek},i})$ where $\mathsf{ct}^{\mathsf{cpa}}_{\mathsf{tag},i}$ is *not* computed by the encryption key ek_i,
- and at the same time, the commitment $c_{\mathsf{ek},i}$ verifies

$$\mathsf{Com.Verify}(\mathsf{pp}^{\mathsf{com}}, c_{\mathsf{ek},i}, \widetilde{\mathsf{ek}_i^{\mathsf{cpa}}}, d_{\mathsf{ek},i}) = 1$$

for some $\widetilde{\mathsf{ek}_i^{\mathsf{cpa}}} \neq \mathsf{ek}_i^{\mathsf{cpa}}$.

This implies that the probability that $\mathsf{Fail}_{\mathsf{tag}^*}$ happens is bounded by the probability that the adversary can break the *soundness* of the NIZK, or the *E2-security* of EECom.
More formally, we simulate the game for a CCA-adversary $\mathcal{A}$ by a simulator, trying to break either the E2-security or the soundness of NIZK[25], as follows:

- $\mathcal{B}$ simulate the crs, then runs the setup algorithm $\mathsf{Setup} \to (\mathsf{msk}, (\mathsf{ek}_i)_i)$, and send to $\mathcal{A}$ the public parameter $\mathsf{pp} := (\lambda, n, \mathsf{crs})$.
- The responses to $\mathcal{A}$ when querying **Extract**, **Corrupt** are simulated using msk and the (statically corrupted) encryption keys $(\mathsf{ek}_i)_i$.
- When $\mathcal{A}$ queries **Enc** with $(x_i^{(j)}, z_i^{(j)} := (\mathsf{tag}, \tilde{z}_i^{(j)}))$, the simulator computes $\mathsf{ct}_i^{(j)} := \mathsf{Enc}(\mathsf{ek}_i, x_i^{(j)}, z_i^{(j)})$ where:
  - $\mathcal{B}$ computes

$$\begin{aligned} r_{\mathsf{tag},i}^{(j)} &\leftarrow \mathsf{PRF}(K_i, x_i^{(j)}, z_i^{(j)}, \mathsf{tag}) \\ (c_{\mathsf{tag},i}^{\mathsf{ee},j}, d_{\mathsf{tag},i}^{\mathsf{ee}}) &\leftarrow \mathsf{EECom.SimCom}(\mathsf{td}, (x_i^{(j)}, z_i^{(j)})) \\ \mathsf{ct}_{\mathsf{tag},i}^{\mathsf{cpa},j} &\leftarrow \mathsf{Enc}^{\mathsf{cpa}}(\mathsf{ek}_i^{\mathsf{cpa}}, x_i^{(j)}, z_i^{(j)}; r_{\mathsf{tag},i}^{(j)}) \ . \end{aligned}$$

  - (If the guess is to break soundness) $\mathcal{B}$ queries its soundness challenger on the statement $(\mathsf{ct}_{\mathsf{tag},i}^{\mathsf{cpa},j}, c_{\mathsf{tag},i}^{\mathsf{ee},j}, c_{\mathsf{ek},i}^{\mathsf{cpa},j}) \in \mathscr{L}_{\mathsf{CtMsgRel}^{\mathsf{sk}}}$. This can be done under an indistinguishable simulated set up $(\overline{\mathsf{crs}}, \mathsf{td}) \leftarrow \mathsf{SimCRS}(1^\lambda)$ and give $\mathcal{B}$ a simulated proof $\pi_{\mathsf{tag},i}$.
  - (Else if the guess is to break the E2-security) $\mathcal{B}$ commits to

$$\mathsf{EECom.Commit}(\mathsf{pp}^{\mathsf{ee}}, (x_i^{(j)}, z_i^{(j)}))$$

    and obtain $(c_{\mathsf{tag},i}^{\mathsf{ee},j}, d_{\mathsf{tag},i}^{\mathsf{ee}})$, then computes the NIZK proof themselves.
  - the ciphertext $\mathsf{ct}_i := (\mathsf{ct}_i^{\mathsf{cpa}}, c_{\mathsf{tag},i}^{\mathsf{ee},j}, \pi_{\mathsf{tag},i})$ is sent to $\mathcal{A}$.
- When $\mathcal{A}$ queries **LoR** with $x_i^{(0,j)}, x_i^{(1,j)}, z_i^{chall,j} := (\mathsf{tag}^*, \tilde{z}_i^{chall,j})$, $\mathcal{B}$ proceeds as above, but with the challenge tag tag^* and in case $\mathcal{B}$ is against the E2-security of EECom, the challenge $c_{\mathsf{tag},i}^{(\mathsf{ee},j,b)}$ is obtained from the simulated commitment procedure

$$\mathsf{EECom.SimCom}(\mathsf{td}, x_i^{(b,j)})$$

  and obtain $(c_{\mathsf{tag},i}^{(\mathsf{ee},j,b)}, d_{\mathsf{tag},i}^{(\mathsf{ee},b)})$
- When $\mathcal{A}$ queries **Dec** with $\mathbf{c} := (\mathsf{ct}_i^{\mathsf{cpa},j}, c_{\mathsf{tag},i}^{\mathsf{ee},j}, \pi_{\mathsf{tag},i}^{(j)})_i$, $\mathcal{B}$ verifies that
  - the proofs $\pi_{\mathsf{tag},i}$ is valid, $\mathsf{Com.Verify}(\mathsf{pp}^{\mathsf{com}}, c_{\mathsf{ek},i}, \mathsf{ek}_i^{\mathsf{cpa}}, d_{\mathsf{ek},i}) = 1$ succeeds for all $i \in [n]$,
  - the extraction $\mathsf{EECom.ExtCom}(\mathsf{td}, c_{\mathsf{tag},i}^{\mathsf{ee},j})$ succeeds for all $i \in \mathcal{C}$.

  If the above verification passes, $\mathcal{B}$
  - runs $(\tilde{x}_i, \tilde{z}_i) \leftarrow \mathsf{EECom.ExtCom}(\mathsf{td}, c_{\mathsf{tag},i}^{\mathsf{ee},j})$ for all $i \in \mathcal{C}$,
  - computes $out \leftarrow F\left(\langle (x_i^{(b,j)}, z_i^{chall,j})_{i\in\mathcal{H}}, (\tilde{x}_i^{(j)}, \tilde{z}_i^{(j)})_{i\in\mathcal{C}} \rangle\right)$,
    where $\langle (x_i^{(b,j)}, z_i^{chall,j})_{i\in\mathcal{H}}, (\tilde{x}_i^{(j)}, \tilde{z}_i^{(j)})_{i\in\mathcal{C}} \rangle$ permutes the elements to their correct ordering as arguments of F,
  - outputs out.

In the end $\mathcal{B}$ outputs the same bit as $\mathcal{A}$. It holds that, when event $\mathsf{Fail}_{\mathsf{tag}^*}$ happens, the **Dec** query for this specific $i \in \mathcal{H}$ helps the adversary to mix different messages in the EECom commitments and the CPA-MCFE ciphertexts, that have *not* been queried to **LoR**, under the same challenge tag tag^*. However, because the adversary $\mathcal{A}$ is admissible (following Definition 29) at the same time, this implies that one of the challenge EECom

[25] This can be formally argued by a sequence of subhybrids where the simulator launches a random guess $d \xleftarrow{\$} \{0,1\}$ and in each case tries to break either the soundness of NIZK or the E2-security of EECom, respectively. The random guess d induces a multiplicative loss of $1/2$ to the advantage of the CCA-adversary, using which the simulator breaks either soundness of NIZK or the E2-security of EECom.

component has been malleated by $\mathcal{A}$[26], or the *simulation soundness* (we recall that the adversary sees many simulated proofs via the encryption queries) of the NIZK is broken[27]. This leads to a union bound on $\Pr[\mathsf{Fail}_{\mathsf{tag}^*}]$ under the soundness of the NIZK and the E2-security of EECom. We conclude that

$$\mathsf{Adv}(\mathsf{G}_1) = \Pr[\mathsf{G}_2 = 1 \mid \neg\mathsf{Fail}_{\mathsf{tag}^*}] \leq \mathsf{Adv}(\mathsf{G}_2) + \Pr[\mathsf{Fail}_{\mathsf{tag}^*}] \leq \mathsf{Adv}(\mathsf{G}_2) + \mathrm{negl}(\lambda)$$

for some negligible function negl and the two games are indistinguishable.

G_3**:** We use the ZK property of NIZK to simulate the proof $\pi_{\mathsf{tag},i}$ in the Enc algorithm. We show below that the change goes indistinguishable under the ZK security of NIZK. More precisely, an adversary $\mathcal{B}$ against the ZK property of NIZK simulates the game for a CCA-adversary $\mathcal{A}$:

- $\mathcal{B}$ receives the crs from its ZK challenger, then runs the setup algorithm $\mathsf{Setup} \to (\mathsf{msk}, (\mathsf{ek}_i)_i)$. Set up the commitment scheme $\mathsf{pp}^{\mathsf{com}} := \mathsf{Com.Setup}(1^\lambda)$ and $\mathsf{pp}^{\mathsf{ee}} := \mathsf{EECom.Setup}(1^\lambda)$. $\mathcal{B}$ then commits to the encryption keys $(c^{\mathsf{cpa}}_{\mathsf{ek},i}, d^{\mathsf{cpa}}_{\mathsf{ek},i}) := \mathsf{Com.Commit}(\mathsf{pp}^{\mathsf{com}}, \mathsf{ek}^{\mathsf{cpa}}_i; r_{\mathsf{ek},i})$, and send to $\mathcal{A}$ the public parameter $\mathsf{pp} := (\lambda, n, \mathsf{crs}, \mathsf{pp}^{\mathsf{com}}, (c^{\mathsf{cpa}}_{\mathsf{ek},i})_i)$.
- The responses to $\mathcal{A}$ when querying **Extract**, **Corrupt** are simulated using msk and (statically demanded up front) the encryption keys $(\mathsf{ek}_i)_i$.
- When $\mathcal{A}$ queries **Enc** with $x^{(j)}_i, z^{(j)}_i$ with $z^{(j)}_i \overset{def}{=} (\mathsf{tag}, \tilde{z}^{(j)}_i)$, the simulator computes $\mathsf{ct}^{(j)}_i := \mathsf{Enc}(\mathsf{ek}_i, x^{(j)}_i, z^{(j)}_i)$ where:
  - $\mathcal{B}$ computes

$$\begin{aligned} r^{(j)}_{\mathsf{tag},i} &\leftarrow \mathsf{PRF}(K_i, x^{(j)}_i, z^{(j)}_i, \mathsf{tag}) \\ (c^{\mathsf{ee},j}_{\mathsf{tag},i}, d^{\mathsf{ee}}_{\mathsf{tag},i}) &\leftarrow \mathsf{EECom.SimCom}(\mathsf{td}, (x^{(j)}_i, z^{(j)}_i)) \\ \mathsf{ct}^{\mathsf{cpa},j}_{\mathsf{tag},i} &\leftarrow \mathsf{Enc}^{\mathsf{cpa}}(\mathsf{ek}^{\mathsf{cpa}}_i, x^{(j)}_i, z^{(j)}_i; r^{(j)}_{\mathsf{tag},i}) \ . \end{aligned}$$

  - $\mathcal{B}$ queries its ZK challenger on the statement $(\mathsf{ct}^{\mathsf{cpa},j}_{\mathsf{tag},i}, c^{\mathsf{ee},j}_{\mathsf{tag},i}, c^{\mathsf{cpa}}_{\mathsf{ek},i}) \in \mathscr{L}_{\mathsf{CtMsgRel}^{\mathsf{sk}}}$, together with a witness $(\mathsf{ek}^{\mathsf{cpa}}_i, r^{(j)}_{\mathsf{tag},i}, d^{\mathsf{ee}}_{\mathsf{tag},i}, (x^{(j)}_i, z^{(j)}_i), d^{\mathsf{cpa}}_{\mathsf{ek},i})$, so as to receive the proof $\pi^{(j)}_{\mathsf{tag},i}$.
  - the ciphertext $\mathsf{ct}^{(j)}_i := (\mathsf{ct}^{\mathsf{cpa},j}_i, c^{\mathsf{ee},j}_{\mathsf{tag},i}, \pi^{(j)}_{\mathsf{tag},i})$ is sent to $\mathcal{A}$.
- When $\mathcal{A}$ queries **Enc** with $x^{(0,j)}_i, x^{(1,j)}_i, z^{chall,j}_i$ with $z^{chall,j}_i \overset{def}{=} (\mathsf{tag}^*, \tilde{z}^{chall,j}_i)$, $\mathcal{B}$ proceeds similarly to the previous case, but with the challenge tag tag^* and the challenge $x^{(b,j)}_i$.
- When $\mathcal{A}$ queries **Dec** with $\mathbf{c} := (\mathsf{ct}^{\mathsf{cpa},j}_i, c^{\mathsf{ee},j}_{\mathsf{tag},i}, \pi^{(j)}_{\mathsf{tag},i})_i$, $\mathcal{B}$ verifies that
  - the proofs $\pi^{(j)}_{\mathsf{tag},i}$ is valid, $\mathsf{Com.Verify}(\mathsf{pp}^{\mathsf{com}}, c_{\mathsf{ek},i}, \mathsf{ek}^{\mathsf{cpa}}_i, d_{\mathsf{ek},i}) = 1$ succeeds for all $i \in [n]$,
  - the extraction $\mathsf{EECom.ExtCom}(\mathsf{td}, c^{\mathsf{ee},j}_{\mathsf{tag},i})$ succeeds for all $i \in \mathcal{C}$.

  From the previous game G_2, conditioned on the E2-security of EECom is not broken nor the simulation soundness of the NIZK, the event $\mathsf{Fail}_{\mathsf{tag}^*}$ does not happen and all **Dec** queries from $\mathcal{A}$ can be answered. If the verification passes, $\mathcal{B}$ runs $\mathsf{Dec}^{\mathsf{cpa}}(\mathsf{dk}_{F_\lambda}, \mathbf{c}^{\mathsf{cpa}} := (\mathsf{ct}^{\mathsf{cpa},j}_i)_i)$.

In the end, $\mathcal{B}$ outputs the same bit as $\mathcal{A}$. If the ZK challenger outputs the real proof, $\mathcal{B}$ is simulating G_2 for $\mathcal{A}$, otherwise the game G_3 is simulated. Hence, the difference in views of $\mathcal{A}$ against theses two games is negligible, bounded under the ZK security of NIZK.

G_4 : We replace the randomness from PRF by uniformly random strings:

$$r^{(j)}_{\mathsf{tag},i} \overset{\$}{\leftarrow} \{0,1\}^{\mathsf{poly}(\lambda)}$$

for some fixed polynomial poly. The randomness is fresh for each i, tag, and also fresh for each repetitionn j. The change is indistinguishable due to the PRF security, combining with the fact that the proofs are simulated independently from the PRF outputs. The latter results from the previous game G_3.

[26] This means either the *Strong Simmulation Indistinguishability* or the *Strong Binding Extractability* of the E2-commitment scheme is broken.

[27] This captures in particular the case where $\mathsf{ct}^{\mathsf{cpa}}_i$ are encrypted under "wild" encryption procedures that do not come from set up, knowing that the commitment $(c^{\mathsf{cpa}}_{\mathsf{ek},i}, d^{\mathsf{cpa}}_{\mathsf{ek},i}) := \mathsf{Com.Commit}(\mathsf{pp}^{\mathsf{com}}, \mathsf{ek}^{\mathsf{cpa}}_i; r_{\mathsf{ek},i})$ perfectly binds ek_i.

G_5 : We switch the challenge ciphertext component (for $i \in \mathcal{H}$ that is known statically) in the EECom commitments to $x_i^{(0,j)}$:

$$\begin{aligned} r_{\mathsf{tag},i}^{(j)} &\xleftarrow{\$} \{0,1\}^{\mathsf{poly}(\lambda)} \\ (c_{\mathsf{tag},i}^{\mathsf{ee},j}, d_{\mathsf{tag},i}^{\mathsf{ee}}) &\leftarrow \mathsf{EECom.SimCom}(\mathsf{td}, (\boxed{x_i^{(0,j)}}, z_i^{chall,j})) \\ \mathsf{ct}_{\mathsf{tag},i}^{\mathsf{cpa},j} &\leftarrow \mathsf{Enc}^{\mathsf{cpa}}(\mathsf{ek}_i^{\mathsf{cpa}}, x_i^{(b,j)}, z_i^{chall,j}; r_{\mathsf{tag},i}^{(j)}) \ . \end{aligned}$$

Thanks to the E2-security of the EECom, the trapdoor that is set up since G_1 gives particularly also its equivocable property (not only the extractability), as well as the simulation soundness of the NIZK, the change is indistinguishable.

1. Our simulator keeps track of the queries from $\mathcal{A}$. As soon as $\mathcal{A}$ submits to **Dec** a query whose components $\left(\widehat{\mathsf{ct}_{\mathsf{tag},i}^{\mathsf{cpa},j}}, \widehat{c_{\mathsf{tag},i}^{\mathsf{ee},j}}\right)$ satisfy *(i)* the commitment $c_{\mathsf{tag},i}^{\mathsf{ee},j}$ of *honest* $i \in \mathcal{H}$ is opened for $x_i^{(0)}$, *(ii)* the MCFE ciphertext is for $x_i^{(b)}$, and *(iii)* the NIZK proof $\pi_{\mathsf{tag},i}^{(j)}$ is valid, the simulator parse the statement $\widehat{stmt}_{\mathsf{nizk}}^{(j)} \stackrel{def}{=} (\widehat{\mathsf{ct}_{\mathsf{tag},i}^{\mathsf{cpa},j}}, \widehat{c_{\mathsf{tag},i}^{\mathsf{ee},j}}, c_{\mathsf{ek},i}^{\mathsf{cpa},j})$ for the relation $\mathsf{CtMsgRel}^{\mathsf{sk}}$ and outputs $(\widehat{stmt}_{\mathsf{nizk}}, \pi_{\mathsf{tag},i}^{(j)})$ to its soundness challenger.
2. At the same time, all encryption queries have the EECom commitments forwarded to the E2-security challenger.

In the end, unless there is a special query that falls into case 1, which breaks the soundness of the NIZK, an adversary $\mathcal{A}$ that distinguishes G_4 from G_5 can be used to break the E2-security of the EECom.

G_6 : Finally, we switch the challenge ciphertext component (for $i \in \mathcal{H}$) in the CPA-secure MCFE ciphertext to $x_i^{(0,j)}$:

$$\begin{aligned} r_{\mathsf{tag},i}^{(j)} &\xleftarrow{\$} \{0,1\}^{\mathsf{poly}(\lambda)} \\ (c_{\mathsf{tag},i}^{\mathsf{ee},j}, d_{\mathsf{tag},i}^{\mathsf{ee}}) &\leftarrow \mathsf{EECom.SimCom}(\mathsf{td}, (x_i^{(0,j)}, z_i^{chall,j})) \\ \mathsf{ct}_{\mathsf{tag},i}^{\mathsf{cpa},j} &\leftarrow \mathsf{Enc}^{\mathsf{cpa}}(\mathsf{ek}_i^{\mathsf{cpa}}, \boxed{x_i^{(0,j)}}, z_i^{chall,j}; r_{\mathsf{tag},i}^{(j)}) \ . \end{aligned}$$

The change is indistinguishable due to the CPA-security of the MCFE.

After G_6 the challenge ciphertext does not depend on $b \xleftarrow{\$} \{0,1\}$ anymore and the advantage of the CCA-adversary is bounded by 1/2. □

H Proof for Section 5.1

Correctness. We first remark that, using ubiquitously the key-homomorphic property of PHF and the fact that $\mathbf{b}_i, \mathbf{b}$ are (self-reduced) yes-instances of $\mathbf{P}$,

$$\sum_{j\in[n]} \mathsf{share}[j,i] = \sum_{j\in[n]} \mathsf{hash}(\delta_{j,i}^{\mathsf{ss}} \cdot \langle \mathbf{y}_i, \mathsf{hk}_j \rangle, \mathbf{b}_i) = \mathsf{hash}\left(\sum_{j\in[n]} \delta_{j,i}^{\mathsf{ss}} \cdot \langle \mathbf{y}_i, \mathsf{hk}_j \rangle, \mathbf{b}_i\right)$$

and

$$\begin{aligned} \mathsf{pt}_i &= \left(\sum_{k\in[N]} \mathbf{y}_{i,k} \cdot \mathbf{c}_{i,k}\right) - \mathsf{hash}(\mathsf{HK}^{\mathsf{ss}}[i], \mathbf{b}) \\ &= \mathsf{hash}\left(\sum_{k\in[N]} \mathbf{y}_{i,k} \cdot \mathsf{hk}_{i,k}, \mathbf{b}_i\right) + \mathsf{hash}\left(\sum_{k\in[N]} \mathbf{y}_{i,k} \cdot \mathsf{hk}_{i,k}, \mathbf{b}\right) + [\![\mathbf{y}_{i,k}\mathbf{x}_{i,k}]\!] - \mathsf{hash}(\langle \mathbf{y}_i, \mathsf{hk}_i + \mathsf{hk}_i^{\mathsf{ss}} \rangle, \mathbf{b}) \\ &= \mathsf{hash}(\langle \mathbf{y}_i, \mathsf{hk}_i \rangle, \mathbf{b}_i) + \mathsf{hash}(\langle \mathbf{y}_i, \mathsf{hk}_i^{\mathsf{ss}} \rangle, \mathbf{b}) + [\![\langle \mathbf{y}_i, \mathbf{x}_i \rangle]\!] \ . \end{aligned}$$

Noticing that using key-homomorphism once more time

$$\begin{aligned}
&\sum_{i=1}^{n}\left(\mathsf{pt}_i + \sum_{j\in[n]} \mathsf{share}[j,i]\right)\\
&= \sum_{i\in[n]}\left(\mathsf{hash}(\langle \mathbf{y}_i, \mathsf{hk}_i\rangle, \mathbf{b}_i) + \mathsf{hash}(\langle \mathbf{y}_i, \mathsf{hk}_i^{\mathsf{ss}}\rangle, \mathbf{b}) + \mathsf{hash}\left(\sum_{j\in[n]} \delta^{\mathsf{ss}}_{j,i}\cdot\langle \mathbf{y}_i, \mathsf{hk}_j\rangle, \mathbf{b}_i\right) + [\![\langle \mathbf{y}_i, \mathbf{x}_i\rangle]\!]\right)\\
&= \sum_{i\in[n]}\left(\mathsf{hash}\left(\langle \mathbf{y}_i, \mathsf{hk}_i\rangle + \sum_{j\in[n]} \delta^{\mathsf{ss}}_{j,i}\cdot\langle \mathbf{y}_i, \mathsf{hk}_j\rangle, \mathbf{b}_i\right) + \mathsf{hash}(\langle \mathbf{y}_i, \mathsf{hk}_i^{\mathsf{ss}}\rangle, \mathbf{b})\right) + \left[\!\!\left[\sum_{i=1}^{n}\langle \mathbf{x}_i, \mathbf{y}_i\rangle\right]\!\!\right]\\
&\overset{(\dagger)}{=} \sum_{i\in[n]} \mathsf{hash}(0, \mathbf{b}_i) + \sum_{i\in[n]} (\mathsf{hash}(\langle \mathbf{y}_i, \mathsf{hk}_i^{\mathsf{ss}}\rangle, \mathbf{b})) + \left[\!\!\left[\sum_{i=1}^{n}\langle \mathbf{x}_i, \mathbf{y}_i\rangle\right]\!\!\right]\\
&\overset{(\dagger)}{=} \sum_{i=1}^{n} \mathsf{hash}(0, \mathbf{b}_i) + \mathsf{hash}\left(\sum_{i\in[n]}\langle \mathbf{y}_i, \mathsf{hk}_i^{\mathsf{ss}}\rangle, \mathbf{b}\right) + \left[\!\!\left[\sum_{i=1}^{n}\langle \mathbf{x}_i, \mathbf{y}_i\rangle\right]\!\!\right]\\
&\overset{(\dagger)}{=} \left(\sum_{i=1}^{n} \mathsf{hash}(0, \mathbf{b}_i)\right) + \mathsf{hash}(0, \mathbf{b}) + \left[\!\!\left[\sum_{i=1}^{n}\langle \mathbf{x}_i, \mathbf{y}_i\rangle\right]\!\!\right]
\end{aligned}$$

where $(\dagger)$ comes from Equation (8) concludes correctness.

Security. We prove the IND-CPA security of the MIFE in Figure 4 in the following theorem. The IND-CPA security notion is recalled in Definition 46, as introduced previously by [44, 46].

Theorem 52. *Let PHF* = (hashkg, projkg, hash, projhash) *be a SPHF for the subset membership problem* $\mathbf{P}$, as per *Definition 5. We suppose further that* $\mathbf{P}$ *is hard and average self-reducible, and PHF is CPA-friendly (following Definition 12). Then the MIFE in Figure 4 for* $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$ *is IND-CPA* as per *Definition 46.*

Proof (Of Theorem 52). Let $\mathbf{P}$ be a subset membership problem whose induced relation is average self-reducible. To recall, the subset membership problem $\mathbf{P}$ is an ensemble of distributions $(I_\lambda)_\lambda$ where for any given λ, the distribution I_λ is efficiently sampleable. We also use a PHF = (hashkg, projkg, hash, projhash) for $\mathbf{P}$ *as per* Definition 5. We require that PHF is *CPA-friendly* as defined in Definition 12. Let PRF = (Keygen, Eval) be a PRF with key space $\mathcal{K}^{\mathsf{prf}}$, domain $\mathcal{D}^{\mathsf{prf}}$, and range $\mathcal{R}^{\mathsf{prf}}$. We consider the MIFE from Figure 4. We recall that for each slot i, the adversary against the IND-CPA security of the MIFE is given access to the encryption oracle $\mathsf{Enc}(\mathsf{ek}_i, \cdot)$, and can query repeatedly $\mathbf{x}_i^{(j)}$'s of its choice, where each repetition is indexed by j that belongs to a polynomially bounded range. In the following we use the superscript (j) to denote the repetition index.

The sequence of games is described below:

G_0. This is the original security experiment for the MIFE with respect to the function class $\mathcal{F}^{\mathsf{IP}}_{\mathsf{subvec},B}$. The advantage of an adversary $\mathcal{A}$ in a game G_i is denoted by $\mathsf{Adv}(\mathsf{G}_i) := |\Pr[\mathsf{G}_i = 1] - 1/2|$ where the probability is taken over the random choices of $\mathcal{A}$ and coins of G_i. The changes that make the transitions between games are highlighted in gray.

G_1. This game is identical to G_0 except that the PRF is replaced by a random function, specifically all random coins

$$r_i^{(j)} \xleftarrow{\$} \boxed{\mathcal{R}}$$

are sampled uniformly at random. Under the PRF security, while noticing that the key K for every ek_i is not leaked to the adversary in the security of MIFE, the transition from G_0 to G_1 is indistinguishable.

G_2. This game is identical to G_1 except that we syntactically rewrite the oracles **LoR, Enc**. Specifically the **LoR** computes as follows: upon the challenge $(\mathbf{x}_{i,k}^{(0,j)}, \mathbf{x}_{i,k}^{(1,j)})_{k\in[N]}$ from the adversary,

$$\begin{aligned}
r_i^{(b,j)} &\xleftarrow{\$} \mathcal{R} \\
\mathbf{b}_i^{(b,j)} &\leftarrow \mathsf{RSR}(\lambda, \mathbf{b}, r_i^{(b,j)}); \quad \omega_i^{(b,j)} \leftarrow \mathsf{RSRw}(\lambda, \mathbf{b}, r_{i,k}^{(b,j)}, \omega) \\
\mathsf{pjk}_{i,k}^{(b,j)} &\leftarrow \mathsf{projkg}(\mathsf{hk}_{i,k}, \mathbf{b}_i^{(b,j)}) \\
\text{for } k \in [N] : \mathbf{c}_{i,k}^{(b,j)} &\stackrel{def}{=} \boxed{\mathsf{hash}(\mathsf{hk}_{i,k}, \mathbf{b}_i^{(b,j)})} + \boxed{\mathsf{hash}(\mathsf{hk}_{i,k}, \mathbf{b})} + \left[\!\left[\mathbf{x}_{i,k}^{(b,j)} \right]\!\right] \in \mathbb{G} \ .
\end{aligned}$$

Define $\mathbf{c}_i^{(b,j)} \stackrel{def}{=} (\mathbf{c}_{i,k}^{(b,j)})_{k=1}^N \in \mathbb{G}^N$ and output $\mathsf{ct}_i^{(b,j)} := (\mathbf{c}_i^{(b,j)}, \mathbf{b}_i^{(b,j)}, \mathbf{b})$. Due to perfect correctness of PHF under (self-reducible) yes-instances, the transition from G_1 to G_2 is perfectly indistinguishable.

G_3. After G_2, the encryption in **(LoR, Enc)** uses hash values instead of projected hash values, thus does not depend on the witness anymore. This allows us to replace the self-randomly reduced instances by uniformly random instances using the efficiently sampleable distribution I_λ of $\mathbf{P}$, and remove the witness from the encryption's process. We denote $D(\mathcal{X}_\lambda)$ the distribution of yes-instances of $\mathbf{P}$, that is specified in I_λ.

$$\begin{aligned}
r_i^{(b,j)} &\xleftarrow{\$} \mathcal{R} \\
\mathbf{b}_i^{(b,j)} &\xleftarrow{\$} \boxed{D(\mathcal{X}_\lambda)} \\
\mathsf{pjk}_{i,k}^{(b,j)} &\leftarrow \mathsf{projkg}(\mathsf{hk}_{i,k}, \mathbf{b}_i^{(b,j)}) \\
\text{for } k \in [N] : \mathbf{c}_{i,k}^{(b,j)} &\stackrel{def}{=} \mathsf{hash}(\mathsf{hk}_{i,k}, \mathbf{b}_i^{(b,j)}) + \mathsf{hash}(\mathsf{hk}_{i,k}, \mathbf{b}) + \left[\!\left[\mathbf{x}_{i,k}^{(b,j)} \right]\!\right] \in \mathbb{G} \ .
\end{aligned}$$

The transition from G_2 to G_3 is statistically indistinguishable thanks to the fact that all random coins are uniformly sampled and the properties of random self-reducibility of $\mathbf{P}$ (*as per* Definition 4).

G_4. We now use the computational hardness of $\mathbf{P}$ to be able to use only no-instance $\widehat{\mathbf{b}} \xleftarrow{\$} D(\mathcal{U}_\lambda \setminus \mathcal{X}_\lambda)$ for encryption process of **LoR, Enc**, as well as simulating the projection keys at setup. We denote $D(\mathcal{X}_\lambda)$ the distribution of yes-instances of $\mathbf{P}$ as well as by $D(\mathcal{U}_\lambda \setminus \mathcal{X}_\lambda)$ of no-instances of $\mathbf{P}$, that are specified in I_λ. The setup makes changes in simulating the projection keys:

$$\forall i \in [n], k \in [N] : \mathsf{hk}_{i,k} \leftarrow \mathsf{hashkg}(\Lambda) \ ; \ \mathsf{pjk}_{i,k} \leftarrow \boxed{\mathsf{projkg}(\mathsf{hk}_{i,k} + \mathsf{hk}_\perp(\widehat{\mathbf{b}}), \widehat{\mathbf{b}})}$$

The encryption in **LoR, Enc** is then computed as follows:

$$\begin{aligned}
r_i^{(b,j)} &\xleftarrow{\$} \mathcal{R} \\
\widehat{\mathbf{b}_i^{(b,j)}} &\xleftarrow{\$} \boxed{D(\mathcal{U}_\lambda \setminus \mathcal{X}_\lambda)} \\
\mathsf{pjk}_{i,k}^{(b,j)} &\leftarrow \mathsf{projkg}(\mathsf{hk}_{i,k}, \boxed{\widehat{\mathbf{b}_i^{(b,j)}}}) \\
\text{for } k \in [N] : \mathbf{c}_{i,k}^{(b,j)} &\stackrel{def}{=} \mathsf{hash}(\mathsf{hk}_{i,k}, \boxed{\widehat{\mathbf{b}_i^{(b,j)}}}) + \mathsf{hash}(\mathsf{hk}_{i,k}, \boxed{\widehat{\mathbf{b}}}) + \left[\!\left[\mathbf{x}_{i,k}^{(b,j)} \right]\!\right] \in \mathbb{G} \ .
\end{aligned}$$

and output $\mathsf{ct}_i^{(j)} := (\mathbf{c}_i^{(b,j)}, \widehat{\mathbf{b}_i^{(b,j)}}, \widehat{\mathbf{b}})$[28]. The transition from G_3 to G_4 is computationally indistinguishable.

1. We apply the hardness of the subset membership problem $\mathbf{P}$. The changes in $\widehat{\mathbf{b}}_i$ is indistinguishable thanks to the computational hardness for distinguishing I_λ-distributed yes-instances $\mathbf{b} \xleftarrow{\$} D(\mathcal{X}_\lambda)$ and no-instances $\widehat{\mathbf{b}_i^{(b,j)}}, \widehat{\mathbf{b}} \xleftarrow{\$} D(\mathcal{U}_\lambda \setminus \mathcal{X}_\lambda)$ from Definition 3.
2. We use CPA-friendliness of PHF (see Definition 12) to switch the instances. In the following $\widehat{\mathbf{b}}$ is a no-instance of $\mathbf{P}$.

[28] Note that we do not use the projection key $\mathsf{pjk}^{(b,j)} \leftarrow \mathsf{projkg}(\mathsf{hk}_{i,k}, \widehat{\mathbf{b}_i^{(b,j)}})$ anymore as all the masks are now hash values.

- $(\mathsf{hk}_\perp, g_\perp, M_\perp)$-strong diversity to be able to simulate the encryption key ek_i that contains the projection keys $\mathsf{pjk}_{i,k}$

$$\mathsf{projkg}(\mathsf{hk}_{i,k} + \mathsf{hk}_\perp(\widehat{\mathbf{b}}), \widehat{\mathbf{b}}) = \mathsf{projkg}(\mathsf{hk}_{i,k}, \widehat{\mathbf{b}})$$

- $(\mathsf{hk}_\perp, M_z, \epsilon_{\mathrm{ti}})$- translation indistinguishability, as well as the property $\mathsf{hk}_\perp(\widehat{\mathbf{b}}) = g_\perp$ when $\widehat{\mathbf{b}}$ is a no-instance ensures the same distribution for challenge ciphertexts. In particular, this applies when we use the no-instances $\widehat{\mathbf{b}}_i^{(b,j)}, \widehat{\mathbf{b}}$ for hashing.

Correctness is ensures thanks to the previous game that we replace all projected hash values by *real* hash values (thus using a no-instance $\widehat{\mathbf{b}}$ does not affect because the ciphertext contains themselves $\widehat{\mathbf{b}}$ for decrypting by rehashing). The translation indistinguishability gives statistical indistinguishability, while the projection keys are perfectly simulated.

By a union bound over two item 1 and item 2, we conclude the computational indisguishability between G_3 and G_4.

G_5. We now use the stronger computational smoothness (Definition 8), given that $\mathbf{P}$ is hard and applying Lemma 37. We emphasize that this point deviates from the simpler case in [18] where smoothness for *random* no-instances suffices. Rather, we required the smoothness for *every* no-instances: this allows protecting against mix-and-match where the adversary can *choose* instances to mix at the time of decryption, given the ciphertexts $\mathsf{ct}_i^{(j)}$ as well as the hash keys from the functional keys, up to repetition j of the ciphertext.

In order to preserve correctness, the functional key queries are simulated below, then we verify that the correctness is preserved.

Setup: We generate an auxiliary hash key

$$\widetilde{\mathsf{hk}} \xleftarrow{\$} \mathsf{hashkg}(\Lambda) \enspace .$$

This key is used only for simulation purpose, along with

$$\forall i \in [n], k \in [N] : \mathsf{hk}_{i,k} \leftarrow \mathsf{hashkg}(\Lambda) \; ; \; \mathsf{pjk}_{i,k} \leftarrow \mathsf{projkg}(\mathsf{hk}_{i,k} + \mathsf{hk}_\perp(\widehat{\mathbf{b}}), \widehat{\mathbf{b}})$$

that result from the previous game.

KeyGen: We simulate the functional key for parameters $(\mathbf{y}_i)_i$ as follows:

- For each $i \in [n]$, we sample random coefficients $\widetilde{\delta^{\mathsf{ss}}_{j,i}} \xleftarrow{\$} \mathbb{Z}_q$, where $j \in [n]$, such that $\sum_{j\in[n]} \widetilde{\delta^{\mathsf{ss}}_{j,i}} \cdot \langle \mathbf{y}_i, \mathsf{hk}_j \rangle = -\widetilde{\mathsf{hk}}$.
- Sample, for each $i \in [n]$, hash keys $\widetilde{\mathsf{hk}^{\mathsf{ss}}_i} \in \mathcal{K}$ that satisfy $\sum_{i\in[n]} \langle \mathbf{y}_i, \mathsf{hk}_i + \widetilde{\mathsf{hk}^{\mathsf{ss}}_i} \rangle = 0$.
- Define $\mathsf{HK}[j,i] \overset{def}{=} \widetilde{\delta^{\mathsf{ss}}_{j,i}} \cdot \langle \mathbf{y}_i, \mathsf{hk}_j \rangle \in \mathcal{K}$, then define $\mathsf{HK}^{\mathsf{ss}}[i] = \langle \mathbf{y}_i, \mathsf{hk}_i + \widetilde{\mathsf{hk}^{\mathsf{ss}}_i} \rangle$.
- And output $\mathsf{dk}_F \overset{def}{=} (\mathsf{HK}, \mathsf{HK}^{\mathsf{ss}})$.

Enc: Next, the computation of the challenge component goes by

$$\widehat{\mathbf{b}_i^{(0,j)}} \xleftarrow{\$} D(\mathcal{U}_\lambda \setminus \mathcal{X}_\lambda)$$

$$\mathbf{c}_{i,k}^{(b,j)} \overset{def}{=} \left[\!\left[\mathbf{x}_{i,k}^{(0,j)} \right]\!\right] + \mathsf{hash}(\widetilde{\mathsf{hk}}, \widehat{\mathbf{b}_i^{(0,j)}}) + [\![\Delta \mathbf{x}_i]\!] \in \mathbb{G} \enspace ,$$

where $\Delta \mathbf{x}_i \overset{def}{=} \mathbf{x}_i^{(b,j)} - \mathbf{x}_i^{(0,j)}$[29]. It is important to note that the random coefficients $\widetilde{\mathsf{hk}}$ are independent from the slot i, the slot i is indexed in the hash values using $\widetilde{\mathsf{hk}}$.

We now verify that the correctness is preserved. First of all,

$$\begin{aligned}
\sum_{j\in[n]} \mathsf{share}[j,i] &= \sum_{j\in[n]} \mathsf{hash}(\widetilde{\delta^{\mathsf{ss}}_{j,i}} \cdot \langle \mathbf{y}_i, \mathsf{hk}_j \rangle, \widehat{\mathbf{b}_i^{(0,j)}}) \\
&= \mathsf{hash}\left(\sum_{j\in[n]} \widetilde{\delta^{\mathsf{ss}}_{j,i}} \cdot \langle \mathbf{y}_i, \mathsf{hk}_j \rangle, \widehat{\mathbf{b}_i^{(0,j)}} \right) \\
&= \mathsf{hash}(-\widetilde{\mathsf{hk}}, \widehat{\mathbf{b}_i^{(0,j)}})
\end{aligned}$$

[29] The value $\Delta \mathbf{x}_i$ is 0 if $b = 0$ and constant due to admissibility if $b = 1$. If it differs following repetitions when $b = 1$, there exists a mix-and-match that allows using decryption to decide b, as we are in MIFE setting.

and

$$\begin{aligned}\mathsf{pt}_i &= \left(\sum_{k\in[N]} \mathbf{y}_{i,k}\cdot \mathbf{c}_{i,k}\right) - \mathsf{hash}(\mathsf{HK}^{\mathsf{ss}}[i], \widehat{\mathbf{b}}) \\ &= \left[\!\left[\langle \mathbf{y}_i, \mathbf{x}_i^{(0,j)}\rangle + \langle \mathbf{y}_i, \Delta\mathbf{x}_i\rangle\right]\!\right] - \mathsf{hash}(\langle \mathbf{y}_i, \mathsf{hk}_i + \widetilde{\mathsf{hk}_i^{\mathsf{ss}}}\rangle, \widehat{\mathbf{b}}) + \sum_{k=1}^{N} \mathsf{hash}(\widetilde{\mathsf{hk}}, \widehat{\mathbf{b}_i^{(0,j)}})\ .\end{aligned}$$

Noticing that using key-homomorphism once more time

$$\begin{aligned}&\sum_{i=1}^{n}\left(\mathsf{pt}_i + \sum_{j\in[n]} \mathsf{share}[j,i]\right) \\ &\overset{(\ddagger)}{=} \sum_{i=1}^{n}\sum_{N}^{k=1} \mathsf{hash}(\widetilde{\mathsf{hk}}, \widehat{\mathbf{b}_i^{(0,j)}}) - \sum_{i=1}^{n} \mathsf{hash}(\langle \mathbf{y}_i, \mathsf{hk}_i + \widetilde{\mathsf{hk}_i^{\mathsf{ss}}}\rangle, \widehat{\mathbf{b}}) + \left[\!\left[\sum_{i=1}^{n}\langle \mathbf{x}_i^{(0,j)}, \mathbf{y}_i\rangle\right]\!\right] \\ &\quad + \left[\!\left[\sum_{i=1}^{n}\langle \mathbf{y}_i, \Delta\mathbf{x}_i\rangle\right]\!\right] - \sum_{i=1}^{n}\sum_{N}^{k=1} \mathsf{hash}(\widetilde{\mathsf{hk}}, \widehat{\mathbf{b}_i^{(0,j)}}) \\ &\overset{(\ddagger)}{=} \sum_{i=1}^{n} \mathsf{hash}(0, \widehat{\mathbf{b}_i^{(0,j)}}) + \mathsf{hash}(0, \widehat{\mathbf{b}}) + \left[\!\left[\sum_{i=1}^{n}\langle \mathbf{x}_i, \mathbf{y}_i\rangle\right]\!\right]\end{aligned}$$

where (‡) comes from the way we simulate $\widetilde{\delta_{j,i}^{\mathsf{ss}}}$ and $\widetilde{\mathsf{hk}_i^{\mathsf{ss}}}$, and the key homomorphism *i.e.* $\mathsf{hash}(-\widehat{\mathsf{hk}}, \cdot) = -\mathsf{hash}(\widehat{\mathsf{hk}}, \cdot)$, together with the admissibility of IND-CPA security for MIFE $\sum_{i=1}^{n}\langle \mathbf{y}_i, \Delta\mathbf{x}_i\rangle = 0$.
The computational smoothness *as per* Definition 8 makes sure $\mathsf{hash}(\mathsf{hk}_i, \mathbf{b}_i^{(b,j)})$ is indistinguishable from a uniformly random element in $\mathbb{G}$. This thus allows using the hash by $\widetilde{\mathsf{hk}}$ on no-instance $\widehat{\mathbf{b}}_i$, whose role as a one-time pad mask then allows switching b to a fixed 0 while the other term $\Delta\mathbf{x}_i$ is constant for a fixed i no matter what repetition j (also coming from the IND-CPA admissibility *vis-à-vis* the repetitions). The transition from G_3 to G_4 is computationally indistinguishable.

The challenge ciphertexts that are received by the adversary in G_4 are independent from the challenge bit b. This makes $\mathsf{Adv}(\mathsf{G}_4) = 0$ and concludes the proof. □