Verifiable Inner Product Encryption Scheme*

Najmeh Soroush[1], Vincenzo Iovino[1,2], Alfredo Rial[1], Peter B. Roenne[1], and Peter Y. A. Ryan[1]

[1] SnT, University of Luxembourg,{firstname.lastname}@uni.lu
[2] University of Salerno, Italy, vinciovino@gmail.com

Abstract. In the standard setting of functional encryption (FE), we assume both the Central Authority (CA) and the encryptors to run their respective algorithms faithfully. Badrinarayanan *et al.* [ASIACRYPT 2016] put forth the concept of verifiable FE, which essentially guarantees that dishonest encryptors and authorities, even when colluding together, are not able to generate ciphertexts and tokens that give "inconsistent" results. They also provide a compiler turning any perfectly correct FE into a verifiable FE, but do not give efficient constructions.
In this paper we improve on this situation by considering Inner-Product Encryption (IPE), which is a special case of functional encryption and a primitive that has attracted wide interest from both practitioners and researchers in the last decade. Specifically, we construct the first *efficient* verifiable IPE (VIPE) scheme according to the inner-product functionality of Katz, Sahai and Waters [EUROCRYPT 2008]. To instantiate the general construction of Badrinarayanan *et al.* we need to solve several additional challenges. In particular, we construct the first efficient *perfectly correct* IPE scheme. Our VIPE satisfies *unconditional* verifiability, whereas its privacy relies on the DLin assumption.

Keywords: Inner-product encryption · verifiability · Functional commitments

1 Introduction

Functional encryption (FE) is a new encryption paradigm that was first proposed by Sahai and Waters [19] and formalized by Boneh, Sahai and Waters [6]. Informally, in an FE system, a decryption key allows a user to learn a *function* of the original message. More specifically, in a FE scheme for functionality $F : K \times \mathcal{M} \to \mathcal{CT}$, defined over *key space* $\mathcal{K}$, *message space* $\mathcal{M}$ and *output space* $\mathcal{CT}$, for every *key* $k \in K$, the owner of the master secret key MSK associated with master public key MPK can generate a token $\mathtt{Tok}_k$ that allows the computation of $F(k, m)$ from a ciphertext of x computed under the master public key MPK.

A notable special case of FE is that of *inner product encryption* (IPE) [7,14,15,18,17]. In IPE the message is a pair $(m, \boldsymbol{x})$, with $m \in \mathcal{M}$, the *payload message* and vector

* The research were Supported by the Luxembourg National Research Fund (FNR) CORE project FESS (no. C16/IS/11299247)

$\boldsymbol{x}$ the *attribute* in a set Σ, and the token is associated with a vector $\boldsymbol{v} \in \Sigma$. The functionality is $F(\boldsymbol{v}, (m, \boldsymbol{x})) = f_{\boldsymbol{v}}(\boldsymbol{x}, m)$ which returns m if $\langle \boldsymbol{x}, \boldsymbol{v} \rangle = 0$ (i.e,. the two vectors are orthogonal) or $\perp$ otherwise. IPE is a generalization of Identity-Based Encryption [20,5,8] and Anonymous Identity-Based Encryption [4,1], and has been the subject of extensive studies in the last decade.

In FE and IPE, the encryptors and the Central Authority (CA) that generate the tokens are assumed to be honest. Indeed, as noticed by Badrinarayanan *et al.*, in presence of any dishonest party (that is, either the party that generates the token or the party who encrypts the message), the decryption outputs may be inconsistent and this raises serious issues in practical applications (e.g., auditing). For instance, a dishonest authority might be able to generate a faulty token $\mathsf{Tok}_{\boldsymbol{v}}$ for a vector $\boldsymbol{v}$ such that $\mathsf{Tok}_{\boldsymbol{v}}$ enables the owner to decrypt a ciphertext for a vector $\boldsymbol{x}$ that is not orthogonal to $\boldsymbol{v}$. Or a dishonest encryptor might generate a faulty ciphertext that decrypts to an incorrect result with an honestly computed token. These issues are particularly severe in the applications to functional commitments that we will see later.

Verifiable Inner Product Encryption (VIPE) overcomes those limitations by adding strong verifiability guarantees to IPE. VIPE is a special case of Verifiable Functional Encryption (VFE), firstly proposed by Badrinarayanan *et al.* [2] for general functionalities. Informally speaking, in VIPE there are public verification algorithms to verify that the output of the setup, encryption and token generation algorithms are computed honestly. Intuitively, if the master public key MPK and a ciphertext CT pass a public verification test, it means there exists some message m and a unique vector $\boldsymbol{x}$ – up to parallelism – such that for all vectors $\boldsymbol{v}$, if a token $\mathsf{Tok}_{\boldsymbol{v}}$ for $\boldsymbol{v}$ is accepted by the verification algorithm then the following holds:

$$\forall \boldsymbol{v} : \mathsf{Dec}(\mathsf{Tok}_{\boldsymbol{v}}, \mathsf{CT}) = f_{\boldsymbol{v}}(\boldsymbol{x}, m)$$

The main component we employ for constructing a VIPE scheme is an IPE scheme. However, it is worth mentioning that most IPE schemes cannot be made verifiable following the general compiler of Badrinarayanan *et al.* because this compiler requires the IPE scheme to have perfect correctness. We will later discuss in depth why this property is crucial in constructing VIPE.

1.1 Our results and applications

Our Contribution. In this paper we construct an efficient VIPE scheme from bilinear maps. Towards this goal, we build a perfectly correct IPE scheme that may be of independent interest. To our knowledge, all IPE schemes known in literature do *not* satisfy perfect correctness. Our perfectly correct IPE scheme is based on standard assumptions over bilinear groups.

We assume the reader to be familiar with the construction of Badrinarayanan *et al.* that transforms a generic FE scheme to a VFE scheme for the same functionality. This transform, for the case of the inner-product functionality of [14], requires a perfectly correct IPE scheme and non-interactive witness-indistinguishable (NIWI) proofs for the relations we will define in Section 5.

Therefore, constructing an efficient VIPE scheme boils down to building an efficient perfectly correct IPE scheme and efficient NIWI proofs for specific relations. The rest of the paper is devoted to achieving these goals.

Motivating applications. IPE has numerous applications, including Anonymous Identity-Based Encryption, Hidden-Vector Encryption, and predicate encryption schemes supporting polynomial evaluation [14]. As shown by Badrinarayanan *et al.* [2], making FE schemes verifiable enables more powerful applications. As an example, in this section we show that VIPE can be used to construct what we call *polynomial commitment scheme* which corresponds to a functional commitment of Badrinarayanan *et al.* for the polynomial evaluation *predicate*. The same construction can easily be adapted to construct functional commitments for the inner-product predicate.

Perfectly binding polynomial commitments. Using a polynomial commitment scheme [13], Alice may publish a commitment to a polynomial $\mathsf{poly}(x)$ with coefficients in $\mathbb{Z}_p$. If later Bob wants to know $\mathsf{poly}(m)$ for some value m, that is the evaluation of the polynomial at some point, he sends m to Alice who replies with the allegedly evaluation y and a proof that $y = \mathsf{poly}(m)$. The proof guarantees that the claimed evaluation is consistent with the committed polynomial. We require the scheme to be *perfectly binding*.

We construct a polynomial commitment scheme for polynomials of degree at most d from a VIPE scheme for vectors of dimension $d + 2$ in the following way. Let $\mathsf{VIP} = \langle \mathsf{VIP.SetUp}, \mathsf{VIP.TokGen}, \mathsf{VIP.Enc}, \mathsf{VIP.Dec} \rangle$ be a VIPE scheme. We define the following algorithms:

- Commitment Phase: To commit to a polynomial $\mathsf{poly}(x) = a_d x^d + a_{d-1}x^{d-1} + \ldots + a_1 x + a_0 \in \mathbb{Z}_p[X]$, run $\mathsf{VIP.SetUp}(1^\lambda, d+2)$ to generate $(\mathsf{MPK}, \mathsf{MSK})$, compute the attribute $\boldsymbol{x} := (a_d, a_{d-1}, \ldots, a_1, a_0, 1) \in \mathbb{Z}_p^{d+2}$ and ciphertext $\mathsf{CT} \rightarrow \mathsf{VIP.Enc}(\mathsf{MPK}, \boldsymbol{x})$, and output the commitment $\mathsf{com} := (\mathsf{MPK}, \mathsf{CT})$.
- Opening phase: In this phase, a party requests a query (m, y) to check if the commitment corresponds to a polynomial poly such that $\mathsf{poly}(m) = y$. The Committer runs the token-generator algorithm of VIP for vector $\boldsymbol{v} := (m^d, m^{d-1}, \ldots, m, 1, -y)$ and sends $\mathsf{Tok}_{\boldsymbol{v}}$ as the opening. Note that $\langle \boldsymbol{x}, \boldsymbol{v} \rangle = a_d m^d + a_{d-1} m^{d-1} + \ldots + a_1 m + a_0 - y = \mathsf{poly}(m) - y$, therefore $\mathsf{VIP.Dec}(\mathsf{CT}, \mathsf{Tok}_{\boldsymbol{v}}) = 0$ iff $\mathsf{poly}(m) = y$

It is straightforward to see that the above algorithms form a functional commitment (in the sense of [2]) for the polynomial evaluation predicate. We defer the reader to [2] for more details on functional commitments.

1.2 Technical overview

To instantiate the transform of Badrinarayanan *et al.* we need to build an IPE scheme with perfect correctness. Our starting point to construct a perfectly correct IPE scheme is the IPE scheme of Park [18] which only enjoys statistical correctness. The reason for choosing this IPE is that it is conceptually simple and its security is based on standard assumptions over bilinear groups. However, to make it perfectly correct, we will need to solve several technical challenges. The main improvements we need to achieve unconditional verifiability are the following:

i. The master public key needs to be verifiable.
ii. The scheme has to satisfy perfect correctness.

This requires substantial modification of all main algorithms: setup, token generation, encryption and decryption.

Verification of algorithm outputs. A VIPE scheme requires public verification algorithms that can verify the outputs of the setup, encryption and token generation algorithms, in particular check whether these algorithms were run honestly. In more detail, if any string (master public key, ciphertext or token) passes the corresponding verification algorithm, it means it was a proper output of the corresponding algorithm (setup, encryption or token generation). Each party who runs the setup, encryption or token generation algorithm needs to provide a proof that it executed the algorithm honestly without revealing harmful information about the secret parameters or the randomness used in the algorithm.

Usually non-interactive Zero-Knowledge (NIZK) proofs are used in this context. Unfortunately, NIZK proofs cannot used for verifiable FE as they rely on a trusted CRS (Common Reference String) or random oracles and we aim at *perfect verifiability* which has to hold despite any collusion and computing power. The transform of Badrinarayanan *et al.* solves the issue by employing NIWI-proofs in a clever way.

Following the transform of [2], our VIPE consists of four parallel IPE. In the VIPE's encryption algorithm we first run each IPE's encryption algorithm to generate four ciphertexts and then we prove that all these four ciphertexts are the encryption of the same message or that some other trapdoor predicate is satisfied (the latter is needed for message indistinguishability and will be detailed later).

For the sake of argument, let us assume the VIPE scheme consists only of two (instead of four) parallel perfectly correct IPE scheme instantiations IP and $\hat{\mathsf{IP}}$. The master public key of the Park's scheme contains a component $\Lambda = e(g, g')$ in which g is public but g' needs to be kept secret. An honestly computed ciphertext CT in IP includes $\mathsf{ct}_1 = g^{-s}$ and $\mathsf{ct}_7 = \Lambda^{-s} \cdot m$ among its components (we here ignore the other components). We first provide proof that CT (resp. $\hat{\mathsf{CT}}$ in $\hat{\mathsf{IP}}$) is well-formed. Then we need to prove that the two ciphertexts are both encryptions of the same message M (i.e., $m = \hat{m} = M$). We reduce the problem to proving that the following property holds:

$$\frac{\mathsf{ct}_7}{\hat{\mathsf{ct}}_7} = \frac{e(g, g')^{-s} \cdot m}{e(\hat{g}, \hat{g}')^{-\hat{s}} \cdot \hat{m}} = \frac{e(\hat{\mathsf{ct}}_1, \hat{g}')}{e(\mathsf{ct}_1, g')} = \frac{e(\hat{g}^{\hat{s}}, \hat{g}')}{e(g^s, g')}$$

However, since g' and $\hat{g}'$ are not public, the party who runs the encryption algorithm would be unable to prove this property. We solve this issue in the following way: We add to the master public key of IP two elements g_1, g_2 (and $\hat{g}_1, \hat{g}_2$ for $\hat{\mathsf{IP}}$) satisfying $\Lambda = e(g, g') = e(g_1, g_2), \hat{\Lambda} = e(\hat{g}, \hat{g}') = e(\hat{g}_1, \hat{g}_2)$. Then, we add the following equations for the new secret variables $\mathcal{X}_3 = g_1^s, \hat{\mathcal{X}}_3 = \hat{g}_1^{\hat{s}}$:

$$\mathsf{ct}_7^{-1} \cdot \hat{\mathsf{ct}}_7 = e(\mathcal{X}_3, g_2) \cdot e(\hat{\mathcal{X}}_3, \hat{g}_2)^{-1},\ e(g, \mathcal{X}_3) = e(\mathsf{ct}_1, g_1), \quad e(\hat{g}, \hat{\mathcal{X}}_3) = e(\hat{\mathsf{ct}}_1, \hat{g}_1)$$

It is easy to see that these equations are satisfied iff $m = \hat{m}$, and now they can be proven by the encryptor. Having modified Park's scheme, we thus have to prove that the modified scheme is IND-secure. This is done in Section 3.1 in which we reduce the IND-Security of the scheme to the Decision Linear assumption.

Achieving perfect correctness. For the Badrinarayanan *et al.*'s transform to work, it is crucial that the underlying IPE scheme have perfect correctness. If the IPE scheme had a negligible probability of decryption error rather than perfect correctness, then dishonest parties might collude with each other so that invalid results would be accepted by the verification algorithms. Contrast this with the aforementioned functional commitments. In the latter primitive, the committer is the same party who generates the ciphertext (the commitment) and the token (the decommitment) and thus might profit from a negligible space of decryption error to prove false assertions on its committed value. To our knowledge, all IPE schemes[1] known in the literature have a negligible probability of error which makes cheating possible and so not directly usable to construct verifiable functional encryption and functional commitments for the IPE functionality.

In more detail, in most pairing-based IPE schemes the encryption and decryption algorithms work as follows:

$$\mathsf{Enc}(\mathsf{MPK}, \boldsymbol{x}, \mathbf{m}) \rightarrow \mathsf{CT}, \;\; \mathsf{Dec}(\mathsf{Tok}_{\boldsymbol{v}}, \mathsf{CT}) \rightarrow m^* = m \cdot (\mathtt{r})^{\langle \boldsymbol{x}, \boldsymbol{v} \rangle},$$

in which $\mathtt{r}$ is some random value that depends on the randomness used by the token generator and encryption algorithms. Thus, even in case of honest parties, there is a negligible probability that $\mathtt{r} = 1$ and so, even if $\langle \boldsymbol{x}, \boldsymbol{v} \rangle \neq 0$, the decryption algorithm may output a valid message $\mathbf{m}$ instead of $\perp$.

In case of dishonest parties, it may happen that two parties (the encryptor and the token generator) collude with each other to create randomness such that $\mathtt{r}$ equals 1. In this case, the parties would be able to provide valid proofs of the fact that they followed the protocol correctly and invalid results would pass the verification algorithms. A similar problem also appears in the context of MPC in the head [12], where the soundness of the ZK protocol built from MPC strongly relies on the perfect correctness of the underlying MPC. To cope with statistical correctness in MPC in the head, a coin tossing protocol can be employed, while in a completely non-interactive scenario like ours this is more challenging. Hence, to obtain a VIPE scheme it is crucial to construct an IPE scheme satisfying perfect correctness.

Recall that the decryption algorithm in the IPE scheme of Park [18] works as follows:

$$\mathsf{Dec}(\mathsf{Tok}_{\boldsymbol{v}}, \mathsf{CT} = \mathsf{Enc}(\boldsymbol{x}, m)) \longrightarrow m^* = m \cdot \mathbf{e}(g, h)^{(\lambda_1 s_3 + \lambda_2 s_4)\langle \boldsymbol{x}, \boldsymbol{v} \rangle}$$

[1] Recall that we refer to the IPE functionality of Katz, Sahai and Waters [14].

in which λ_1, λ_2 are randomn values used in the token generation algorithm and s_3, s_4 are random values used in the encryption algorithm. To decide whether to accept the output of the decryption or not, the first attempt would be the following. Generate two ciphertexts $\mathsf{ct}, \mathsf{ct}'$ with two independent random values $\{s_i\}, \{s'_i\}$, decrypt both ct and ct' to get M and M' and if $M = M'$ accept the result, or output $\perp$ otherwise. In more detail:

$$M = m \cdot e(h,g)^{(\lambda_1 s_3 + s_4 \lambda_2)\langle \boldsymbol{x}, \boldsymbol{v} \rangle}, M' = m \cdot e(h,g)^{(\lambda_1 s'_3 + s'_4 \lambda_2)\langle \boldsymbol{x}, \boldsymbol{v} \rangle}$$

However, in case $\langle \boldsymbol{x}, \boldsymbol{v} \rangle \neq 0$ there is non-zero probability for which:

$$\lambda_1 s_3 + s_4 \lambda_2 = \lambda_1 s'_3 + \lambda_2 s'_4 \neq 0 \Rightarrow M = M' \neq m$$

To avoid this issue, we choose the random values in such a way that the above equality can never occur. To do so, in the encryption algorithm we choose non-zero random values $s_1, \ldots, s_4$ and $s'_1, \ldots, s'_4$ such that $s_3 \neq s'_3$, and $s_4 = s'_4$. In this case, we have:

$$\lambda_1 s_3 + s_4 \lambda_2 = \lambda_1 s'_3 + \lambda_2 s_4 \Rightarrow \lambda_1 (s_3 - s'_3) = 0 \Rightarrow (\lambda_1 = 0) \vee (s_3 = s'_3)$$

Based on the way λ_1, s_3, s'_3 have been chosen, neither $(\lambda_1 = 0)$ nor $(s_3 = s'_3)$ may happen, hence the decryption algorithm outputs m if and only if $\langle \boldsymbol{x}, \boldsymbol{v} \rangle = 0$. The resulting IPE scheme satisfies perfect correctness as wished. We will prove that the new IPE scheme is still selectively indistinguishability-secure. When constructing a VIPE scheme from such IPE scheme, these additional constraints in the encryption and token generation procedures will correspond to more constraints in the proofs of correct encryption and token generation.

Furthermore, an additional challenge we will have to address is that some of the proofs in the Badrinarayanan*et al.*transform are for relations that consist of a generalized form of distjunction and thus standard techniques to implement disjunctions for GS proofs cannot be directly applied, see Section 5.1.

1.3 Related work and comparison

Verifiable functional encryption has been introduced by Badrinarayanan *et al.* [2], who provide a construction for general functionalities.

Recently, [3] introduced a new FE scheme that supports an extension of the inner-product functionality. The scheme is perfectly correct assuming the message space to be short. However, notice that when employing the scheme in order to construct an IPE scheme (according to the functionality of Katz, Sahai and Waters [14]) the perfect correctness is *lost.* In essence, the IPE constructed from the scheme in [3] would encrypt some additional random value r so that the decryption would return the value $m + r \cdot \langle \boldsymbol{x}, \boldsymbol{v} \rangle$. In this way, if the vectors $\boldsymbol{x}$ and $\boldsymbol{v}$ are orthogonal then the payload message m is obtained, otherwise a random value is returned.

As corollary of our VIPE, we obtain functional commitments (in the sense of [2]) for the polynomial evaluation and inner-product *predicate.* A similar form of commitments has been proposed by Libert *et al.* [16] but differs from ours in different aspects. In the Libert *et al.*'s scheme, the decommitter reveals the evaluations of the inner-product of the committed vector with any vector of its choice, whereas in ours just the binary value of the inner-product predicate (i.e whether the two vectors are orthogonal or not) is leaked. Our functional commitments are perfectly binding rather than computational binding as in Libert *et al.* Moreover, ours are not based on any trust assumption, whereas in [16] the generator of the public-key can completely break the binding property.

Roadmap. In Section 2 we provide the building blocks and the basic terminology used in this paper. In section 3 we construct our perfectly correct IPE scheme and prove its security based on the Decisional Bilinear Diffie-Hellman and DLin assumptions. In Section 4 we define VIPE and present one candidate construction built on perfectly correct IPE and the NIWI proofs of Section 5.

2 Preliminaries

Notation. Throughout the paper, we use $\lambda \in \mathbb{N}$ as a security parameter. For any integer $n > 0$, we denote by $[n]$ the set $\{1, \ldots, n\}$. PPT stands for probabilistic polynomial time algorithm and $\mathsf{negl}(\lambda)$ denotes a negligible function in λ.

2.1 Building blocks

Definition 1 (Bilinear map [5]). *A bilinear map consists of a pair of groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p endowed with a map $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ satisfying:*

1. *Bilinearity: for all $a, b \in \mathbf{Z}$, $e(g^a, g^b) = e(g, g)^{ab}$ for any $g \in \mathbb{G}$.*
2. *Non-degeneracy: $e(g, g) \neq 1_{\mathbb{G}_T}$ for any $g \in \mathbb{G}$.*
3. *Computability: there exists an efficient algorithm to compute the map.*

Definition 2 (NIWI). *A non-interactive witness indistinguishable proof system (NIWI) is a pair of PPT algorithms $\langle \mathcal{P}, \mathcal{V} \rangle$ for a NP-relation R_L satisfying the following properties:*

1. *Completeness: for all $(x, w) \in R_L, \Pr[\, \mathcal{V}(x, \pi) = 1 \mid \pi \longleftarrow \mathcal{P}(x, w)\,] = 1$.*
2. *Perfect soundness: for every $x \notin L$ and $\pi \in \{0,1\}^*$, $\Pr[\, \mathcal{V}(x, \pi) = 1\,] = 0$.*
3. *Witness indistinguishability: for any sequence $\{(x_n, w_{1,n}, w_{2,n})\}_{n \in \mathbb{N}}$, which $x_n \in \{0,1\}^n, w_{1,n}, w_{2,n} \in R_L(x_n)$, the following holds:*
   $n \in \mathbb{N} : \{\pi_{1,n} |\ \pi_{1,n} \leftarrow \mathcal{P}(x_n, w_{1,n})\}_n \approx_c \{\pi_{2,n} |\ \pi_{2,n} \leftarrow \mathcal{P}(x_n, w_{2,n})\}_n$.

*Groth and Sahai (GS) [11] provide NIWI systems for the satisfiability of what they call "**Pairing Products Equations**" that can be used to instantiate the relations needed in our VIPE construction (cf. Construction 7). Using the techniques of [10], such proofs may be made perfectly sound.*

IPE Scheme: For any $n > 0$, let Σ_n be a set of vectors of length n defined over some field and let $\mathcal{M}$ be a message space. For any vector $\boldsymbol{v} \in \Sigma_n$, the function $f_{\boldsymbol{v}} : \Sigma_n \times \mathcal{M} \to \mathcal{M} \cup \{\perp\}$ is

$$f_{\boldsymbol{v}}(\boldsymbol{x}, m) = \begin{cases} m \text{ If } \langle \boldsymbol{x}, \boldsymbol{v} \rangle = 0 \\ \perp \text{ If } \langle \boldsymbol{x}, \boldsymbol{v} \rangle \neq 0 \end{cases} .$$

Both $\mathcal{M}$, n and the field size can depend on the security parameter λ but for simplicity hereafter we will skip this detail. IPE can be seen as a FE scheme for the previous functionality. More concretely, an IPE scheme is defined as follows.

Definition 3 (IPE Scheme). *An IPE scheme* IP *for a message space* $\mathcal{M}$ *and for a family of sets* $\Sigma = \{\Sigma_n\}_{n>0}$ *consisting of sets of vectors of length* n *over some field is a tuple of four PPT algorithms* $\mathsf{IP} = \langle \mathsf{IP.SetUp}, \mathsf{IP.TokGen}, \mathsf{IP.Enc}, \mathsf{IP.Dec} \rangle$ *with the following syntax and satisfying the following correctness property.*

- $\mathsf{IP.SetUp}(1^\lambda, n) \longrightarrow (\mathsf{MPK}, \mathsf{MSK})$*: the setup algorithm, on input the security parameter* λ *and the vector length* n*, generates master public key* MPK *and master secret key* MSK *for that parameter.*
- $\mathsf{IP.TokGen}(\mathsf{MPK}, \mathsf{MSK}, \boldsymbol{v}) \longrightarrow \mathsf{Tok}_{\boldsymbol{v}}$*: on input master keys and vector* $\boldsymbol{v} \in \Sigma_n$*, the token generation algorithm generates the token* $\mathsf{Tok}_{\boldsymbol{v}}$
- $\mathsf{IP.Enc}(\mathsf{MPK}, \overrightarrow{x}, m) \longrightarrow \mathsf{CT}$*: the encryption algorithm encrypts message* $m \in \mathcal{M}$ *and vector* $\boldsymbol{x} \in \Sigma_n$ *under the master public key.*
- $\mathsf{IP.Dec}(\mathsf{MPK}, \mathsf{Tok}_{\boldsymbol{v}}, \mathsf{CT}) \longrightarrow m' \in \mathcal{M} \cup \{\perp\}$.
- Perfect correctness: IP *is perfectly correct if for all* $\lambda, n > 0, \boldsymbol{x}, \boldsymbol{v} \in \Sigma_n$ *and all* $m \in \mathcal{M}$ *the following holds:*

$$\Pr\left[\mathsf{Dec}(\mathsf{MPK}, \mathsf{Tok}_{\boldsymbol{v}}, \mathsf{CT}) = f_{\boldsymbol{v}}(\boldsymbol{x}, m) \mid \begin{array}{l} (\mathsf{MPK}, \mathsf{MSK}) \longleftarrow \mathsf{IP.SetUp}(1^\lambda, n), \\ \mathsf{Tok}_{\boldsymbol{v}} \longleftarrow \mathsf{IP.TokGen}(\mathsf{MSK}, \boldsymbol{v}), \\ \mathsf{CT} \longleftarrow \mathsf{VIP.Enc}(\mathsf{MPK}, \boldsymbol{x}, m) \end{array} \right] = 1$$

The selectively indistinguishability-based notion of security for an IPE scheme over the vector space Σ and message space $\mathcal{M}$ is formalized by means of the game $\mathsf{IND}^{\mathcal{A},\mathcal{C},\lambda,n}$ in Fig 1, between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ (defined in the game) parameterized by security parameter λ and dimension n. The advantage of $\mathcal{A}$ in this game is $\mathsf{Adv}_{\mathsf{IP},\lambda,n}(\mathcal{A}) = \left| \Pr\left[\mathsf{IND}^{\mathcal{A},\mathsf{IP},\lambda,n} = 1 \right] - \frac{1}{2} \right|$.

Definition 4. *An IPE scheme* IP *is selectively-indistinguishable secure (*IND*-Secure) if for all* $n > 0$ *and all PPT adversaries* $\mathcal{A}$*,* $\mathsf{Adv}_{\mathsf{IP},\lambda,n}(\mathcal{A})$ *is a negligible function of* λ*.*

2.2 Hardness assumptions

We conjecture that the following problems hold relative to some bilinear group generator $\mathsf{GroupGen}(1^\lambda) \longrightarrow (p, \mathbb{G}, \mathbb{G}_T, e)$ that takes security parameter λ as input and outputs λ-bit prime p, the descriptions of two groups $\mathbb{G}$ and $\mathbb{G}_T$ of order p and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

- **Selective Challenge Phase.** $\mathcal{A}(1^\lambda, n) \longrightarrow \boldsymbol{x}_0, \boldsymbol{x}_1 \in \Sigma_n$. Then $\mathcal{A}$ sends these two vectors to the challenger.
- **Setup Phase.** The challenger generates the pair $(\mathsf{MSK}, \mathsf{MPK})$ by invoking the setup algorithm on input $(1^\lambda, n)$. Then $\mathcal{C}$ sends MPK to $\mathcal{A}$.
- **Query Phase 1.** $\mathcal{A}$ asks for the token for a vector $\boldsymbol{v_i} \in \Sigma_n$.
- **Challenge Phase.** $\mathcal{A}$ sends to the challenger two messages $m_0, m_1 \in \mathcal{M}$ of the same length.
- $\mathcal{C}$ flips a coin to generate random bit b and send $\mathsf{CT} = \mathsf{Enc}(\mathsf{MPK}, \boldsymbol{x_b}, m_b)$.
- **Query Phase 2.** Query Phase 2: same as Query Phase 1.
- **Output Phase.** $\mathcal{A}$ outputs a bit b'.
- **Winning Condition.** $\mathcal{A}$ wins the game if $b' = b$ and the following condition is met. It is required that if $m_0 \neq m_1$, $\langle \boldsymbol{x}_0, \boldsymbol{v}_i \rangle, \langle \boldsymbol{x}_1, \boldsymbol{v}_i \rangle \neq 0$ for all the vectors $\boldsymbol{v}_i$ queried in both query phase 1 and 2, or $\langle \boldsymbol{v}_i, \boldsymbol{x}_0 \rangle = 0$ iff $\langle \boldsymbol{v}_i, \boldsymbol{x}_1 \rangle = 0$ otherwise. If the winning condition is satisfied the output of the game is 1 or 0 otherwise.

Fig. 1. Security Game $\mathsf{IND}^{\mathcal{A},\mathsf{IP},\lambda,n}$

Assumption 1 *The Decisional Bilinear Diffie-Hellman assumption (DBDH) in bilinear groups $(p, \mathbb{G}, \mathbb{G}_T, e)$ states the hardness for PPT adversaries of solving the following problem. On input $(g, g^\alpha, g^\beta, g^\gamma, Z) \in \mathbb{G}^4 \times \mathbb{G}_T$, decide whether $Z = e(g, g^{\alpha\beta\gamma})$ or it is a random element in $\mathbb{G}_T$.*

Assumption 2 *The Decisional Linear assumption (DLin) in a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e)$ states the hardness for PPT adversaries of solving the following problem. On input $(g, g^\alpha, g^\beta, g^{\alpha\tau}, g^{\beta\eta}, Z) \in \mathbb{G}^6$, decide whether $Z = g^{\eta+\tau}$ or a random element in $\mathbb{G}$.*

In this paper we use the following equivalent formulation of DLin given in [18]: on input $(g, g^\alpha, g^\beta, g^\tau, g^{\alpha\eta}, Z) \in \mathbb{G}^6$ decide whether $Z = g^{\beta(\eta+\tau)}$ or a random element.

Note that DLin is stronger than DBDH. In the rest of this paper we assume the existence of a bilinear group generator $\mathsf{GroupGen}$ such that DLin (and thus DBDH) holds relative to it.

3 Our perfectly correct inner-product encryption

In this section we construct our perfectly correct IPE, the key ingredient for building verifiable inner-product encryption (see Section 4).

Let $\mathsf{GroupGen}(1^\lambda) \longrightarrow (p, \mathbb{G}, \mathbb{G}_T, e)$ be a bilinear group generator, and $n \in \mathbb{N}$ be the vector length. We construct a perfectly correct IPE scheme $\mathsf{IP} = \langle \mathsf{IP.SetUp}, \mathsf{IP.Enc}, \mathsf{IP.TokGen}, \mathsf{IP.Dec} \rangle$ for the set $\mathbb{Z}_p^n$ of vectors of length n over $\mathbb{Z}_p$ and for message space $\mathcal{M} = \mathbb{G}_T$.

Construction 1 [Our perfectly correct IPE scheme IP]

- $\mathsf{IP.SetUp}(1^\lambda, n) \longrightarrow (\mathsf{MSK}, \mathsf{MPK})$:
  For security parameter λ, $i \in [n]$ and $b \in [2]$, compute what follows:

1. Run $\mathsf{GroupGen}(1^\lambda)$ (cf. Section 2.2) to generate a tuple $\langle p, \mathbb{G}, \mathbb{G}_T, e\rangle$.
2. Pick $g, g' \leftarrow \mathbb{G}$ and $\delta_1, \theta_1, \delta_2, \theta_2, w_{1,i}, t_{1,i}, f_{b,i}, h_{b,i}, k \leftarrow \mathbb{Z}_p^*$
3. Pick $\Omega \leftarrow \mathbb{Z}_p$ and compute $\{w_{2,i}, t_{2,i}\}_{i\in[n]}$ such that:
$$\Omega = \delta_1 w_{2,i} - \delta_2 w_{1,i} = \theta_1 t_{2,i} - \theta_2 t_{1,i}.$$
4. For $i = 1, \dots n, b = 1, 2$, set:
$$W_{b,i} = g^{w_{b,i}}, \quad F_{b,i} = g^{f_{b,i}}, \quad K_1 = g^k, \quad U_b = g^{\delta_b}, \quad h = g^{\Omega},$$
$$T_{b,i} = g^{t_{b,i}}, \quad H_{b,i} = g^{h_{b,i}}, \quad K_2 = g'^{\frac{1}{k}}, \quad V_b = g^{\theta_b}, \quad \Lambda = \mathbf{e}(g, g').$$
5. Set:
$$\mathsf{MPK} = [(p, \mathbb{G}, \mathbb{G}_t, e), (g, h, \{W_{b,i}, F_{b,i}, T_{b,i}, H_{b,i}, U_b, V_b\}_{b\in[2], i\in[n]}, K_1, K_2, \Lambda) \in \mathbb{G}^{8n+8} \times \mathbb{G}_T)],$$
$$\mathsf{MSK} = (\{w_{b,i}, f_{b,i}, t_{b,i}, h_{b,i}, \delta_b, \theta_b\}_{b\in[2], i\in[n]}, g') \in \mathbb{Z}_p^{8n+4} \times \mathbb{G}.$$
6. Return $(\mathsf{MPK}, \mathsf{MSK})$.

– $\mathsf{IP.Enc}(\mathsf{MPK}, \boldsymbol{x}, m) \longrightarrow \mathsf{CT}$:
1. For $\boldsymbol{x} = (x_1, \dots, x_n) \in \mathbb{Z}_p^n$ and a message $m \in \mathbb{G}_T$, pick random elements $s_1, \dots s_4, s'_1, \dots, s'_3 \leftarrow \mathbb{Z}_p^*$ such that $s_3 \neq s'_3$ and compute what follows:
$$\mathsf{ct}_1 = g^{s_2},\ \mathsf{ct}_2 = h^{s_1}, \begin{Bmatrix} \mathsf{ct}_{3,i} = W_{1,i}^{s_1} \cdot F_{1,i}^{s_2} \cdot U_1^{x_i s_3} \ , \mathsf{ct}_{4,i} = W_{2,i}^{s_1} \cdot F_{2,i}^{s_2} \cdot U_2^{x_i s_3} \\ \\ \mathsf{ct}_{5,i} = T_{1,i}^{s_1} \cdot H_{1,i}^{s_2} \cdot V_1^{x_i s_4} \ , \mathsf{ct}_{6,i} = T_{2,i}^{s_1} \cdot H_{2,i}^{s_2} \cdot V_2^{x_i s_4} \end{Bmatrix}_{i\in[n]},$$
$$\mathsf{ct}_7 = \mathbf{e}(g^{s_3}, g^{s_4}),\ \mathsf{ct}_8 = \Lambda^{-s_2} \cdot m.$$
$$\mathsf{ct}'_1 = g^{s'_2},\ \mathsf{ct}'_2 = h^{s'_1}, \begin{Bmatrix} \mathsf{ct}'_{3,i} = W_{1,i}^{s'_1} \cdot F_{1,i}^{s'_2} \cdot U_1^{x_i s'_3} \ , \mathsf{ct}'_{4,i} = W_{2,i}^{s'_1} \cdot F_{2,i}^{s'_2} \cdot U_2^{x_i s'_3} \\ \\ \mathsf{ct}'_{5,i} = T_{1,i}^{s'_1} \cdot H_{1,i}^{s'_2} \cdot V_1^{x_i s_4} \ , \mathsf{ct}'_{6,i} = T_{2,i}^{s'_1} \cdot H_{2,i}^{s'_2} \cdot V_2^{x_i s_4} \end{Bmatrix}_{i\in[n]},$$
$$\mathsf{ct}'_7 = \mathbf{e}(g^{s'_3}, g^{s_4}),\ \mathsf{ct}'_8 = \Lambda^{-s'_2} \cdot m.$$
2. Set:
$$\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \begin{Bmatrix} \mathsf{ct}_{3,i} \ , \mathsf{ct}_{4,i} \\ \mathsf{ct}_{5,i} \ , \mathsf{ct}_{6,i} \end{Bmatrix}, \mathsf{ct}_7, \mathsf{ct}_8)\ ,\ \mathsf{ct}' = (\mathsf{ct}'_1, \mathsf{ct}'_2, \begin{Bmatrix} \mathsf{ct}'_{3,i} \ , \mathsf{ct}'_{4,i} \\ \mathsf{ct}'_{5,i} \ , \mathsf{ct}'_{6,i} \end{Bmatrix}, \mathsf{ct}'_7, \mathsf{ct}'_8).$$
3. Output $\mathsf{CT} = (\mathsf{ct}, \mathsf{ct}')$.

– $\mathsf{IP.TokGen}(\mathsf{MSK}, \boldsymbol{v}) \longrightarrow \mathsf{Tok}_{\boldsymbol{v}}$:
1. Pick $\lambda_1, \lambda_2 \leftarrow \mathbb{Z}_p^*$ and for any $i \in [n]$ pick $\{r_i\}\ ,\ \{\Phi_i\} \leftarrow \mathbb{Z}_p^*$.
2. Set $\mathsf{Tok}_{\boldsymbol{v}} = (K_A, K_B, \begin{Bmatrix} K_{3,i} \ , K_{4,i} \\ K_{5,i} \ , K_{6,i} \end{Bmatrix}_{i\in[n]})$ as follows and return $\mathsf{Tok}_{\boldsymbol{v}}$.
$$K_A = g' \cdot \prod_{i=1}^{n} K_{3,i}^{-f_{1,i}} \cdot K_{4,i}^{-f_{2,i}} \cdot K_{5,i}^{-h_{1,i}} \cdot K_{6,i}^{-h_{2,i}}, \qquad K_B = \prod_{i=1}^{n} g^{-(r_i + \Phi_i)}.$$
$$K_{3,i} = g^{-\delta_2 r_i} \cdot g^{\lambda_1 v_i w_{2,i}}, \qquad K_{4,i} = g^{\delta_1 r_i} \cdot g^{-\lambda_1 v_i w_{1,i}}.$$
$$K_{5,i} = g^{-\theta_2 \Phi_i} \cdot g^{\lambda_2 v_i t_{2,i}}, \qquad K_{6,i} = g^{\theta 1 \Phi_i} \cdot g^{-\lambda_2 v_i t_{1,i}}.$$

– $\mathsf{IP.Dec}(\mathsf{CT}, \mathsf{Tok}_{\boldsymbol{v}})$:
Let $\mathsf{CT} = (\mathsf{ct}, \mathsf{ct}')$, such that $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \{\mathsf{ct}_{3,i}, \mathsf{ct}_{4,i}, \mathsf{ct}_{5,i}, \mathsf{ct}_{6,i}\}, \mathsf{ct}_7, \mathsf{ct}_8)$, $\mathsf{ct}' = (\mathsf{ct}'_1, \mathsf{ct}'_2, \{\mathsf{ct}'_{3,i}, \mathsf{ct}'_{4,i}, \mathsf{ct}'_{5,i}, \mathsf{ct}'_{6,i}\}, \mathsf{ct}_7, \mathsf{ct}_8)$
  1. If $\mathsf{ct}_7 = \mathsf{ct}'_7$ output $\perp$ and stop, otherwise go to the next step.
  2. Compute:

$$\varUpsilon = \mathsf{ct}_8 \cdot \mathbf{e}(\mathsf{ct}_1, K_A) \cdot \mathbf{e}(\mathsf{ct}_2, K_B) \cdot \prod_{i=1}^{n} \mathbf{e}(\mathsf{ct}_{3,i}, K_{3,i}) \cdot \mathbf{e}(\mathsf{ct}_{4,i}, K_{4,i}) \cdot \mathbf{e}(\mathsf{ct}_{5,i}, K_{5,i}) \cdot \mathbf{e}(\mathsf{ct}_{6,i}, K_{6,i}).$$

$$\varUpsilon' = \mathsf{ct}'_8 \cdot \mathbf{e}(\mathsf{ct}'_1, K_A) \cdot \mathbf{e}(\mathsf{ct}'_2, K_B) \cdot \prod_{i=1}^{n} \mathbf{e}(\mathsf{ct}'_{3,i}, K_{3,i}) \cdot \mathbf{e}(\mathsf{ct}'_{4,i}, K_{4,i}) \cdot \mathbf{e}(\mathsf{ct}'_{5,i}, K_{5,i}) \cdot \mathbf{e}(\mathsf{ct}'_{6,i}, K_{6,i}).$$

  3. If $\varUpsilon = \varUpsilon'$ output $\varUpsilon$ otherwise output $\perp$.

Perfect Correctness: We now show that an honestly generated ciphertext decrypts correctly with probability 1. Since $F_{1,i}^{-s_2} \cdot \mathsf{ct}_{3,i} = W_{1,i}^{s_1} \cdot U_1^{s_3 x_i}$, we get

$$\mathbf{e}(F_{1,i}^{-s_2} \cdot \mathsf{ct}_{3,i}, K_{3,i}) = \mathbf{e}(g,g)^{s_1\lambda_1 v_i w_{1,i} w_{2,i} - s_3 x_i \delta_1 \delta_2} \cdot \mathbf{e}(g,g)^{-s_1 r_i \delta_2 w_{1,i} + s_3 \lambda_1 v_i \delta_1 w_{2,i}}$$

$$\mathbf{e}(F_{2,i}^{-s_2} \cdot \mathsf{ct}_{4,i}, K_{4,i}) = \mathbf{e}(g,g)^{-s_1\lambda_1 v_i w_{1,i} w_{2,i} + s_3 x_i \delta_1 \delta_2} \cdot \mathbf{e}(g,g)^{s_1 r_i \delta_1 w_{2,i} - s_3 \lambda_1 v_i \delta_2 w_{1,i}}$$

We then get

$$\begin{aligned}\mathbf{e}(F_{1,i}^{-s_2} \cdot \mathsf{ct}_{3,i}, K_{3,i}) \cdot \mathbf{e}(F_{2,i}^{-s_2} \cdot \mathsf{ct}_{4,i}, K_{4,i}) &= \Big(\mathbf{e}(g^{s_1}, g^{r_i}) \cdot \mathbf{e}(g^{x_i s_3}, g^{\lambda_1 v_i})\Big)^{\delta_1 w_{2,i} - \delta_2 w_{1,i}} \\ &= \mathbf{e}(h^{s_1}, g^{r_i}) \cdot \mathbf{e}(h^{s_3 \lambda_1}, g^{x_i v_i}) = \mathbf{e}(\mathsf{ct}_2, g^{r_i}) \cdot \mathbf{e}(h^{\lambda_1 s_3}, g^{x_i v_i})\end{aligned}$$

The same computation gives us

$$\mathbf{e}(H_{1,i}^{-s_2} \cdot \mathsf{ct}_{5,i}, K_{5,i}) \cdot \mathbf{e}(H_{2,i}^{-s_2} \cdot \mathsf{ct}_{6,i}, K_{6,i}) = \mathbf{e}(\mathsf{ct}_2, g^{\varPhi_i}) \cdot \mathbf{e}(h^{\lambda_2 s_4}, g^{x_i v_i})$$

As a conclusion we have the following:

$$\begin{aligned}&\mathbf{e}(\mathsf{ct}_1, K_A) \cdot \prod_{i=1}^{n} \mathbf{e}(\mathsf{ct}_{3,i}, K_{3,i}) \cdot \mathbf{e}(\mathsf{ct}_{4,i}, K_{4,i}) \cdot \mathbf{e}(\mathsf{ct}_{5,i}, K_{5,i}) \cdot \mathbf{e}(\mathsf{ct}_{6,i}, K_{6,i}) = \\ &= \varLambda^{s_2} \prod_{i=1}^{n} \mathbf{e}(F_{1,i}^{-s_2}, K_{3,i}) \mathbf{e}(F_{1,i}^{-s_2}, K_{4,i}) \cdot \mathbf{e}(H_{1,i}^{-s_2}, K_{5,i}) \cdot \mathbf{e}(H_{1,i}^{-s_2}, K_{6,i}) = \\ &= \varLambda^{s_2} \cdot \mathbf{e}(\mathsf{ct}_2, K_B^{-1}) \cdot \mathbf{e}(h,g)^{(\lambda_1 s_3 + \lambda_2 s_4)\langle \boldsymbol{x}, \boldsymbol{v}\rangle}\end{aligned}$$

Plugging this into the decryption algorithm we get

$$\varUpsilon = m \cdot \mathbf{e}(h,g)^{(\lambda_1 s_3 + \lambda_2 s_4)\langle \boldsymbol{x}, \boldsymbol{v}\rangle}, \ \varUpsilon' = m \cdot \mathbf{e}(h,g)^{(\lambda_1 s'_3 + s_4 \lambda_2)\langle \boldsymbol{x}, \boldsymbol{v}\rangle}$$

First note that it cannot happen that $\mathsf{ct}_7 \neq \mathsf{ct}'_7$ for honestly generated ciphertexts. Clearly, $\langle \boldsymbol{x}, \boldsymbol{v}\rangle = 0 \Rightarrow (\varUpsilon = \varUpsilon' = m)$. All we need to check is thus that if $\langle \boldsymbol{x}, \boldsymbol{v}\rangle \neq 0$, we get output $\perp$. We could only get a wrong output if it happens that $\varUpsilon = \varUpsilon'$, but this is impossible since it implies (using $\lambda_1 \neq 0, s_3 \neq s'_3$)

$$\mathbf{e}(h,g)^{(\lambda_1 s_3 - \lambda_1 s'_3)\langle \boldsymbol{x}, \boldsymbol{v}\rangle} = 1_{\mathbb{G}_T} \Rightarrow \lambda_1(s_3 - s'_3)\langle \boldsymbol{x}, \boldsymbol{v}\rangle \equiv_p 0 \ \Rightarrow \langle \boldsymbol{x}, \boldsymbol{v}\rangle \equiv_p 0 \ .$$

3.1 Security reduction to DLin and DBDH

In this section we prove our IPE scheme is IND-Secure under the standard computational assumptions.

Theorem 1. *The IPE scheme* IP *of Construction 1 is* IND*-Secure if the DBDH and DLin assumptions hold relative to* $\mathsf{GroupGen}$*.*

To prove the theorem we define a series of hybrid experiments $\mathsf{H}_0, \ldots, \mathsf{H}_{12}$ in which H_0 corresponds to the real experiment with challenge bit $b = 0$ and H_{12} corresponds to the real experiment with challenge bit $b = 1$, and we show that they are computationally indistinguishable.

- **Hybrid** H_0: this hybrid is identical to the real game with challenge bit $b = 0$. Precisely, the ciphertext is computed for message m_0 and vector $\boldsymbol{x}$ as follows:

$$\mathsf{ct} = (g^{s_2}, h^{s_1}, \{W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{x_i s_3}, T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{x_i s_4}\}_{b\in[2], i\in[n]}, \mathbf{e}(g^{s_3}, g^{s_4}), \Lambda^{-s_2} \cdot m_0)$$

$$\mathsf{ct}' = (g^{s'_2}, h^{s'_1}, \{W_{b,i}^{s'_1} \cdot F_{b,i}^{s'_2} \cdot U_b^{x_i s'_3}, T_{b,i}^{s'_1} \cdot H_{b,i}^{s'_2} \cdot V_b^{x_i s_4}\}_{b\in[2], i\in[n]}, \mathbf{e}(g^{s'_3}, g^{s_4}), \Lambda^{-s'_2} \cdot m_0)$$

- **Hybrid** H_1: this hybrid is identical to the previous hybrid except that instead of $\mathbf{e}(g, g)^{s_3 s_4}, \mathbf{e}(g, g)^{s'_3 s_4}$, the ciphertext contains two random elements $R_1, R'_1 \leftarrow \mathbb{G}_T$. Precisely, the ciphertext is computed as follows:

$$\mathsf{ct} = (g^{s_2}, h^{s_1}\{W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{x_i s_3}, T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{x_i s_4}\}_{b\in[2], i\in[n]}, \boxed{R_1}, \Lambda^{-s_2} \cdot m_0)$$

$$\mathsf{ct}' = (g^{s'_2}, h^{s'_1}, \{W_{b,i}^{s'_1} \cdot F_{b,i}^{s'_2} \cdot U_b^{x_i s'_3}, T_{b,i}^{s'_1} \cdot H_{b,i}^{s'_2} \cdot V_b^{x_i s_4}\}_{b\in[2], i\in[n]}, , \boxed{R'_1}, \Lambda^{-s'_2} \cdot m_0)$$

- **Hybrid** H_2: this hybrid is identical to the previous hybrid except that instead of $\Lambda^{-s_2} \cdot m_0, \Lambda^{-s'_2} \cdot m_0$, the ciphertext contains two random elements $R, R' \leftarrow \mathbb{G}_T$. Precisely, the ciphertext is computed as follows:

$$\mathsf{ct} = (g^{s_2}, h^{s_1}, \{W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{x_i s_3}, T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{x_i s_4}\}_{b\in[2], i\in[n]}, R_1, \boxed{R})$$

$$\mathsf{ct}' = (g^{s'_2}, h^{s'_1}, \{W_{b,i}^{s'_1} \cdot F_{b,i}^{s'_2} \cdot U_b^{x_i s'_3}, T_{b,i}^{s'_1} \cdot H_{b,i}^{s'_2} \cdot V_b^{x_i s_4}\}_{b\in[2], i\in[n]}, , R'_1, \boxed{R'})$$

- **Hybrid** H_3: this hybrid is identical to the previous hybrid except that instead of $T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{x_i s_4}, T_{b,i}^{s'_1} \cdot H_{b,i}^{s'_2} \cdot V_b^{x_i s_4}$, the ciphertext contains $T_{b,i}^{s_1} \cdot H_{b,i}^{s_2}, T_{b,i}^{s'_1} \cdot H_{b,i}^{s'_2}$. Precisely, the ciphertext is computed as follows:

$$\mathsf{ct} = (g^{s_2}, h^{s_1}\{W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{x_i s_3}, \boxed{T_{b,i}^{s_1} \cdot H_{b,i}^{s_2}}\}_{b\in[2], i\in[n]}, R_1, R)$$

$$\mathsf{ct}' = (g^{s'_2}, h^{s'_1}\{W_{b,i}^{s'_1} \cdot F_{b,i}^{s'_2} \cdot U_b^{x_i s'_3}, \boxed{T_{b,i}^{s'_1} \cdot H_{b,i}^{s'_2}}\}_{b\in[2], i\in[n]}, R'_1, R')$$

- **Hybrid** H_4: this hybrid is identical to the previous hybrid except that instead of $T_{b,i}^{s_1} \cdot H_{b,i}^{s_2}, T_{b,i}^{s'_1} \cdot H_{b,i}^{s'_2}$, the ciphertext contains $T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{y_i s_4}, T_{b,i}^{s'_1} \cdot H_{b,i}^{s'_2} \cdot V_b^{y_i s_4}$. Precisely, the ciphertext is computed as follows:

$$\mathsf{ct} = (g^{s_2}, h^{s_1}\{, W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{x_i s_3}, \boxed{T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{y_i s_4}}\}_{b\in[2], i\in[n]}, R_1, R)$$

$$\mathsf{ct}' = (g^{s'_2}, h^{s'_1}\{W_{b,i}^{s'_1} \cdot F_{b,i}^{s'_2} \cdot U_b^{x_i s'_3}, \boxed{T_{b,i}^{s'_1} \cdot H_{b,i}^{s'_2} \cdot V_b^{y_i s_4}}\}_{b\in[2], i\in[n]}, R'_1, R')$$

- **Hybrid** H_5: $\mathsf{CT}_6 = (\mathsf{ct}, \mathsf{ct}')$, This hybrid is identical to the previous hybrid except that the power of V_b in ct is s_4 and its power in ct' is s'_4. Precisely, the ciphertext is computed as follows:

$$\mathsf{ct} = (g^{s_2}, h^{s_1}\{, W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{x_i s_3}, T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{y_i s_4}\}_{b\in[2], i\in[n]}, R_1, R)$$

$$\mathsf{ct}' = (g^{s'_2}, h^{s'_1}\{W_{b,i}^{s'_1} \cdot F_{b,i}^{s'_2} \cdot U_b^{x_i s'_3}, T_{b,i}^{s'_1} \cdot H_{b,i}^{s'_2} \cdot V_b^{y_i s'_4}\}_{b\in[2], i\in[n]}, R'_1, R')$$

- **Hybrid** H_6: this hybrid is identical to the previous hybrid except that $s_3 = s'_3$. Precisely:

$$\mathsf{ct} = (g^{s_2}, h^{s_1}\{W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{x_i s_3}, T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{y_i s_4}\}_{b\in[2], i\in[n]}, R_1, R)$$

$$\mathsf{ct}' = (g^{s'_2}, h^{s'_1}\{W_{b,i}^{s'_1} \cdot F_{b,i}^{s'_2} \cdot U_b^{x_i s_3}, T_{b,i}^{s'_1} \cdot H_{b,i}^{s'_2} \cdot V_b^{y_i s'_4}\}_{b\in[2], i\in[n]}, R'_1, R')$$

- **Hybrid** H_7: This hybrid is identical to the previous hybrid except we replace s_3 with 0.

$$\mathsf{ct} = (g^{s_2}, h^{s_1}\{W_{b,i}^{s_1} \cdot F_{b,i}^{s_2}, T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{y_i s_4}\}_{b\in[2], i\in[n]}, R_1, R)$$

$$\mathsf{ct}' = (g^{s'_2}, h^{s'_1}\{W_{b,i}^{s'_1} \cdot F_{b,i}^{s'_2}, T_{b,i}^{s'_1} \cdot H_{b,i}^{s'_2} \cdot V_b^{y_i s'_4}\}_{b\in[2], i\in[n]}, R'_1, R')$$

- **Hybrid** H_8: This hybrid is identical to the previous hybrid except that instead of $W_{b,i}^{s_1} \cdot F_{b,i}^{s_2}, W_{b,i}^{s'_1} \cdot F_{b,i}^{s'_2}$, we set $W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{y_i s_3}, W_{b,i}^{s'_1} \cdot F_{b,i}^{s'_2} \cdot U_b^{y_i s_3}$. Precisely:

$$\mathsf{ct} = (g^{s_2}, h^{s_1}\{W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{y_i s_3}, T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{y_i s_4}\}_{b\in[2], i\in[n]}, R_1, R)$$

$$\mathsf{ct}' = (g^{s'_2}, h^{s'_1}\{W_{b,i}^{s'_1} \cdot F_{b,i}^{s'_2} \cdot U_b^{y_i s_3}, T_{b,i}^{s'_1} \cdot H_{b,i}^{s'_2} \cdot V_b^{y_i s'_4}\}_{b\in[2], i\in[n]}, R'_1, R')$$

- **Hybrid** H_9: this hybrid is identical to the previous hybrid except that instead of $W_{b,i}^{s_1} \cdot F_{b,i}^{s_2}, W_{b,i}^{s'_1} \cdot F_{b,i}^{s'_2}$, we set $W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{y_i s_3}, W_{b,i}^{s'_1} \cdot F_{b,i}^{s'_2} \cdot U_b^{y_i s'_3}$. Precisely:

$$\mathsf{ct} = (g^{s_2}, h^{s_1}\{W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{y_i s_3}, T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{y_i s_4}\}_{b\in[2], i\in[n]}, R_1, R)$$

$$\mathsf{ct}' = (g^{s'_2}, h^{s'_1}\{W_{b,i}^{s'_1} \cdot F_{b,i}^{s'_2} \cdot U_b^{y_i s'_3}, T_{b,i}^{s'_1} \cdot H_{b,i}^{s'_2} \cdot V_b^{y_i s'_4}\}_{b\in[2], i\in[n]}, R'_1, R')$$

- **Hybrid** H_{10}: this hybrid is identical to the previous hybrid except that instead of $W_{b,i}^{s_1} \cdot F_{b,i}^{s_2}, W_{b,i}^{s'_1} \cdot F_{b,i}^{s'_2}$, we set $W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{y_i s_3}, W_{b,i}^{s'_1} \cdot F_{b,i}^{s'_2} \cdot U_b^{y_i s'_3}$. Precisely:

$$\mathsf{ct} = (g^{s_2}, h^{s_1}\{W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{y_i s_3}, T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{y_i s_4}\}_{b\in[2], i\in[n]}, R_1, R)$$

$$\mathsf{ct}' = (g^{s'_2}, h^{s'_1}\{W_{b,i}^{s'_1} \cdot F_{b,i}^{s'_2} \cdot U_b^{y_i s'_3}, T_{b,i}^{s'_1} \cdot H_{b,i}^{s'_2} \cdot V_b^{y_i s_4}\}_{b\in[2], i\in[n]}, R'_1, R')$$

- **Hybrid** H_{11}: this hybrid is identical to the previous hybrid except that instead of choosing $R, R' \leftarrow \mathbb{G}_T$, we set $R = \Lambda^{-s_2} \cdot m_1, R' = \Lambda^{-s'_2} \cdot m_1$. Precisely, the ciphertext is computed as follows:

$$\mathsf{ct} = (g^{s_2}, h^{s_1}\{W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{y_i s_3}, T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{y_i s_4}\}_{b\in[2], i\in[n]}, R_1, \Lambda^{-s_2} \cdot m_1)$$

$$\mathsf{ct}' = (g^{s'_2}, h^{s'_1}\{W_{b,i}^{s'_1} \cdot F_{b,i}^{s'_2} \cdot U_b^{y_i s'_3}, T_{b,i}^{s'_1} \cdot H_{b,i}^{s'_2} \cdot V_b^{y_i s_4}\}_{b\in[2], i\in[n]}, R'_1, \Lambda^{-s'_2} \cdot m_1)$$

● **Hybrid** H_{12}: this hybrid is identical to the previous hybrid except that instead of R_1, R_1', we set $\mathbf{e}(g^{s_3}, g^{s_4}), \mathbf{e}(g^{s_3'}, g^{s_4})$, which is identical to the real game with challenge bit $b = 1$, in particular for message m_1 and vector $\boldsymbol{y}$. Precisely, the ciphertext is computed as follows:

$$\mathsf{ct} = (g^{s_2}, h^{s_1}\{W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{y_i s_3}, T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{y_i s_4}\}_{b\in[2],i\in[n]}, \mathbf{e}(g^{s_3}, g^{s_4}), \Lambda^{-s_2} \cdot m_1)$$

$$\mathsf{ct}' = (g^{s_2'}, h^{s_1'}\{W_{b,i}^{s_1'} \cdot F_{b,i}^{s_2'} \cdot U_b^{y_i s_3'}, T_{b,i}^{s_1'} \cdot H_{b,i}^{s_2'} \cdot V_b^{y_i s_4}\}_{b\in[2],i\in[n]}, \mathbf{e}(g^{s_3'}, g^{s_4}), \Lambda^{-s_2'} \cdot m_1)$$

Proposition 2 *If the DLin assumption holds relative to* GroupGen*, then* H_0 *and* H_1 *are computationally indistinguishable.*

Proof. Let us assume there exists a PPT adversary $\mathcal{A}$ which distinguishes between H_i and H_{i+1} with non-negligible advantage ϵ. We describe a simulator $\mathcal{B}$ which uses $\mathcal{A}$, on input $(g, A = g^{\alpha}, B = g^{\beta}, C = g^{\tau}, D = g^{\alpha\eta}, Z) \in \mathbb{G}^6$, output 1 if $Z = g^{\beta(\eta+\tau)}$ and 0 if Z is a random element in $\mathbb{G}$. $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

SetUp phase. The adversary $\mathcal{A}$ sends to the simulator, $\mathcal{B}$, two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_p^n$. The simulator picks $g' \leftarrow \mathbb{G}$ and $\tilde{\Omega}, k, \tilde{\delta}_b, \theta_b, \{w_{1,i}, \tilde{t}_{1,i}, f_{b,i}, h_{b,i}\}_{i\in[n],b\in[2]} \leftarrow \mathbb{Z}_p$, compute $\{w_{2,i}, \tilde{t}_{2,i}\}_{i\in[n]}$ such that for each i, $\tilde{\Omega} = \tilde{\delta}_1 w_{2,i} - \tilde{\delta}_2 w_{1,i} = \theta_1 \tilde{t}_{2,i} - \theta_2 \tilde{t}_{1,i}$. Compute the master public key components as follows and returns it:

$$\{W_{b,i} = g^{w_{b,i}}, F_{b,i} = g^{f_{b,i}}\}_{b\in[2],i\in[n]}, \{U_b = A^{\tilde{\delta}_b}\}_{b\in[2]}, h = A^{\tilde{\Omega}}, \Lambda = \mathbf{e}(g, g').$$
$$\{T_{b,i} = A^{\tilde{t}_{b,i}}, H_{b,i} = g^{h_{b,i}}\}_{b\in[2],i\in[n]}, \{V_b = g^{\theta_b}\}_{b\in[2]}, K_1 = g^k,\ K_2 = g'^{\frac{1}{k}}.$$

By doing so, $\mathcal{B}$ implicitly sets $\delta_b = \alpha\tilde{\delta}_b, t_{b,i} = \alpha\tilde{t}_{b,i}$ for $b \in [2], i \in [n]$ and $\Omega = \alpha\tilde{\Omega}$, which shows that each element of the master public key is independently and uniformly distributed in $\mathbb{Z}_p$. Also notice that for each $i \in [n]$, we have: $\delta_1 w_{2,i} - \delta_2 w_{1,i} = \alpha\tilde{\delta}_1 w_{2,i} - \alpha\tilde{\delta}_2 w_{1,i} = \theta_1 \alpha\tilde{t}_{2,i} - \theta_2 \alpha\tilde{t}_{1,i} = \theta_1 t_{2,i} - \theta_2 t_{1,i} = \alpha\tilde{\Omega} = \Omega$. hence the output has the same structure as output by the real setup algorithm.

Token query phase. $\mathcal{B}$ knows all the secret parameters except $\{\delta_b, t_{b,i}\}_{b\in[2],i\in[n]}, \Omega$. When $\mathcal{A}$ asks for a query for a vector $\boldsymbol{v}$, $\mathcal{B}$ picks $\lambda_1, \tilde{\lambda}_2, \{\tilde{r}_i, \Phi_i\}_{i\in[n]} \leftarrow \mathbb{Z}_p^{\star}$. In generating $\mathsf{Tok}_{\boldsymbol{v}}$, the simulator implicitly sets, $\lambda_2 = \alpha\tilde{\lambda}_2, r_i = \alpha\tilde{r}_i$ which are independently and uniformly distributed in $\mathbb{Z}_p^{\star}$. Token elements are set as follows:

$$K_{3,i} = A^{-\tilde{\delta}_2 r_i} \cdot g^{\lambda_1 v_i w_{2,i} x_i} = \text{ (by the above settings) } = g^{-\delta_2 r_i} \cdot g^{v_i w_{2,i} \lambda_1}.$$
$$K_{5,i} = g^{-\theta_2 \phi_i} \cdot A^{\lambda_2 v_i \tilde{t}_{2,i} x_i} = \text{(by the above settings)} = g^{-\theta_2 \phi_i} \cdot g^{\lambda_2 v_i t_{2,i} x_i}.$$
Similarly, $K_{4,i} = A^{\tilde{\delta}_1 r_i} \cdot g^{-\lambda_1 v_i w_{1,i} x_i}, K_{6,i} = g^{\theta_1 r_i} \cdot A^{-\lambda_2 v_i \tilde{t}_{1,i} x_i}$.
$$K_B = \prod_{i=1}^{n} A^{-r_i} g^{-\Phi_i} = \prod_{i=1}^{n} g^{-(\alpha\tilde{r}_i + \Phi_i)} = \prod_{i=1}^{n} g^{-(r_i + \Phi_i)}.$$

$\mathcal{B}$ knows $\{f_{b,i}, h_{b,i}\}_{b\in[2],i\in[n]}$, hence it can compute K_A.

Generating the challenge ciphertext. $\mathcal{A}$ sends message m_0 to $\mathcal{B}$. To generate a challenge ciphertext, $\mathcal{B}$ picks $s_1, s_2, s'_1, s'_2, \tilde{s}_3, \tilde{s}_4, \tilde{s}'_3 \leftarrow \mathbb{Z}_p^\star$ such that $\tilde{s}_3 \neq \tilde{s}'_3$. $\mathcal{B}$ implicitly sets $s_3 = \eta\tilde{s}_3, s_4 = \beta\tilde{s}_4$ and computes the ciphertext as follows:

$$\begin{array}{ll}
\mathsf{ct}_1 = g^{s_2}, \mathsf{ct}'_1 = g^{s'_2} & , \mathsf{ct}_2 = h^{s_1}, \mathsf{ct}'_2 = h^{s'_1}. \\
\mathsf{ct}_{3,i} = W_{1,i}^{s_1} \cdot F_{1,i}^{s_2} \cdot D^{\tilde{\delta}_1 \tilde{s}_3 x_i} & , \mathsf{ct}'_{3,i} = W_{1,i}^{s'_1} \cdot F_{1,i}^{s'_2} \cdot D^{\tilde{\delta}_1 x_i \tilde{s}'_3}. \\
\mathsf{ct}_{4,i} = W_{2,i}^{s_1} \cdot F_{2,i}^{s_2} \cdot D^{\tilde{\delta}_2 \tilde{s}_3 x_i} & , \mathsf{ct}'_{4,i} = W_{2,i}^{s'_1} \cdot F_{2,i}^{s'_2} \cdot D^{\tilde{\delta}_2 x_i \tilde{s}'_3}. \\
\mathsf{ct}_{5,i} = T_{1,i}^{s_1} \cdot H_{1,i}^{s_2} \cdot B^{\theta_1 \tilde{s}_4 x_i} & , \mathsf{ct}'_{5,i} = T_{1,i}^{s'_1} \cdot H_{1,i}^{s'_2} \cdot B^{\theta_1 \tilde{s}_4 x_i}. \\
\mathsf{ct}_{6,i} = T_{2,i}^{s_1} \cdot H_{2,i}^{s_2} \cdot B^{\theta_2 \tilde{s}_4 x_i} & , \mathsf{ct}'_{6,i} = T_{2,i}^{s'_1} \cdot H_{2,i}^{s'_2} \cdot B^{\theta_2 \tilde{s}_4 x_i}. \\
\mathsf{ct}_7 = (\frac{\mathbf{e}(Z,g)}{\mathbf{e}(B,C)})^{\tilde{s}_3 \tilde{s}_4}, \mathsf{ct}_8 = \mathbf{e}(g, g')^{-s_2} \cdot m_0 & , \mathsf{ct}'_7 = (\frac{\mathbf{e}(Z,g)}{\mathbf{e}(B,C)})^{\tilde{s}'_3 \tilde{s}_4}, \mathsf{ct}'_8 = \mathbf{e}(g, g')^{-s'_2} \cdot m_0
\end{array}$$

Since, $D^{\tilde{\delta}_b x_i \tilde{s}_3} = g^{\alpha\tilde{\delta}_b \eta \tilde{s}_3 x_i} = U_b^{x_i s_3}, B^{\theta_b \tilde{s}_4 x_i} = V_1^{\beta \tilde{s}_4 x_i} = V_b^{s_4 x_i}$, by this settings, for each $i \in [n]$, $\mathsf{ct}_{3,i}, \mathsf{ct}'_{3,i}, \ldots, \mathsf{ct}_{6,i}, \mathsf{ct}'_{6,i}$ are computed properly.

Analysing the game: There are two cases, $Z = g^{\beta(\tau+\eta)}$ or $Z \leftarrow \mathbb{G}$:

- $Z = g^{\beta(\tau+\eta)} \Rightarrow \dfrac{\mathbf{e}(Z,g)}{\mathbf{e}(B,C)} = \dfrac{\mathbf{e}(g^{\beta(\tau+\eta)}, g)}{\mathbf{e}(g^\beta, g^\tau)} = \dfrac{\mathbf{e}(g^\beta, g^\tau) \cdot \mathbf{e}(g^\beta, g^\eta)}{\mathbf{e}(g^\beta, g^\tau)} = \mathbf{e}(g^\eta, g^\beta)$

  $\Rightarrow \mathsf{ct}_7 = (\dfrac{\mathbf{e}(Z,g)}{\mathbf{e}(B,C)})^{\tilde{s}_3 \tilde{s}_4} = \mathbf{e}(g^{\eta\tilde{s}_3}, g^{\beta\tilde{s}_4}) = \mathbf{e}(g^{s_3}, g^{s_4}), \mathsf{ct}'_7 = \mathbf{e}(g^{s'_3}, g^{s_4})$

  $\Rightarrow \mathcal{A}$ interacting with H_0.
- $Z \leftarrow \mathbb{G} \Rightarrow \mathsf{ct}_7, \mathsf{ct}'_7$ random elements in $\mathbb{G}_T \Rightarrow \mathcal{A}$ interacting with H_1.

□

Proposition 3 *If the DBDH assumption holds relative to* GroupGen*, then* H_1 *and* H_2 *are computationally indistinguishable.*

Proof. The proof of this proposition is similar to the proof of Lemma 5 in [18] with a slight modification in the SetUp phase. In our scheme, the master public key has elements, K_1 and K_2, such that $e(K_1, K_2) = \Lambda$. In Park's paper, to generate the master public key, $\mathcal{B}$ randomly choose $\tilde{a}$ and sets $\Lambda = \mathbf{e}(A, B)^{-\Omega} \cdot \mathbf{e}(A, g)^{\tilde{a}}$, this imply that $g' = g^{-\alpha\beta\Omega + \alpha\tilde{a}}$. In our scheme, $\mathcal{B}$ sets $K_1 = B^{-k\Omega} \cdot g^{\tilde{a}k}$ and $K_2 = A^{\frac{1}{k}}$, for some random $k \leftarrow \mathbb{Z}_p^\star$. The following computation shows that the master public key is well-formed:

$$\mathbf{e}(K_1, K_2) = \mathbf{e}(A^k, B^{\frac{-\Omega}{k}} \cdot g^{\frac{\tilde{a}}{k}}) = \mathbf{e}(g^\alpha, g^{-\Omega\beta} \cdot g^{\tilde{a}}) = \mathbf{e}(g^{-\alpha\beta\Omega + \alpha\tilde{a}}, g) = \Lambda. \quad □$$

Proposition 4 *If the DLin assumption holds relative to* GroupGen*, then* H_2 *and* H_3 *are computationally indistinguishable.*

The proof of the previous proposition is identical to the corresponding proof in Park [18] and we omit it.

Proposition 5 *If the DLin assumption holds relative to* GroupGen*, then* H_3 *and* H_4 *are computationally indistinguishable.*

Proof. The simulator takes as input $(g, A = g^\alpha, B = g^\beta, C = g^\tau, D = g^{\alpha\eta}, Z \stackrel{?}{=} g^{\beta(\eta+\tau)})$ and by interacting with the adversary $\mathcal{A}$, distinguish between $g^{\beta(\eta+\tau)}$ and a random element.

SetUp phase. Generating master public key is same as in 4, except for $i \in [n], b \in [2]$ it implicitly defines $t_{b,i} = \beta\theta_b y_i + \alpha\tilde{t}_{b,i}, h_{b,i} = \beta\theta_b y_i + \tilde{h}_{b,i}$ and sets $T_{b,i} = B^{\theta_b y_i} A^{\tilde{t}_{b,i}}, H_{b,i} = B^{\theta_b y_i} g^{\tilde{h}_{b,i}}$ The proof that the simulator generates a master public key distributed as the one created by the real challenger is identical to the proof for proposition 4.

Token query phase. The simulator chooses $\tilde{\lambda}_1, \tilde{\lambda}_2, \{\tilde{r}_i, \tilde{\Phi}_i\}_{i\in[n]} \leftarrow \mathbb{Z}_p^\star$, and then implicitly defines the following random values:

$$\lambda_1 = \tilde{\lambda}_1 - \frac{c_y\tilde{\lambda}_2}{\alpha}\ ,\ \lambda_2 = \frac{c_x\tilde{\lambda}_2}{\alpha}\ ,\ r_i = \tilde{r}_i - \frac{c_y\beta v_i x_i\tilde{\lambda}_2}{\alpha}\ ,\ \Phi_i = \tilde{\Phi}_i + \frac{c_x\beta v_i y_i\tilde{\lambda}_2}{\alpha}$$

First, observe that:

$$\begin{aligned}
&-\delta_2 r_i + \lambda_1 v_i w_{2,i} = -\delta_2(\tilde{r}_i - \frac{c_y\beta v_i x_i\tilde{\lambda}_2}{\alpha}) + (\tilde{\lambda}_1 - \frac{c_y\tilde{\lambda}_2}{\alpha})v_i w_{2,i}\\
&= -\delta_2\tilde{r}_i + \frac{c_y\tilde{\lambda}_2}{\alpha}v_i(\underbrace{\beta x_i - w_{2,i}}_{\alpha\tilde{w}_{2,i}}) + \tilde{\lambda}_1 v_i w_{2,i} = -\delta_2\tilde{r}_i + c_y\tilde{\lambda}_2 v_i\tilde{w}_{2,i} + \tilde{\lambda}_1 v_i w_{2,i} =\\
&\Rightarrow K_{3,i} = g^{-\delta_2\tilde{r}_i}\cdot g^{c_y\tilde{\lambda}_2 v_i\tilde{w}_{2,i}}\cdot W_{2,i}^{\tilde{\lambda}_1 v_i} \Rightarrow K_{3,i} \text{ is computable}
\end{aligned}$$

Similar computation:

$$K_{4,i} = g^{\delta_1\tilde{r}_i}\cdot g^{-c_y\tilde{\lambda}_2 v_i\tilde{w}_{1,i}}\cdot W_{1,i}^{-\tilde{\lambda}_1 v_i}, K_{5,i} = g^{-\theta_2\tilde{\Phi}_i}\cdot g^{c_x\tilde{\lambda}_2 v_i\tilde{t}_{2,i}}, K_{6,i} = g^{\theta_1\tilde{\Phi}_i}\cdot g^{-c_x\tilde{\lambda}_2 v_i\tilde{t}_{1,i}}$$

$$\begin{aligned}
K_A &= g'\cdot\prod_{i=1}^{n} K_{3,i}^{-f_{1,i}} K_{4,i}^{-f_{2,i}} K_{5,i}^{-h_{1,i}} K_{6,i}^{-h_{2,i}} = g'\cdot\prod_{i=1}^{n} K_{3,i}^{-f_{1,i}} K_{4,i}^{-f_{2,i}} K_{5,i}^{-\beta\theta_1 y_i - \tilde{h}_{1,i}} K_{6,i}^{-\beta\theta_2 y_i + \tilde{h}_{2,i}}\\
&= g'\cdot\prod_{i=1}^{n} K_{3,i}^{-\ f_{1,i}} K_{4,i}^{-f_{2,i}} K_{5,i}^{-\tilde{h}_{1,i}} K_{6,i}^{-\tilde{h}_{2,i}} (\cancel{g^{-\theta_2\Phi_i}} g^{\lambda_2 v_i t_{2,i}})^{-\beta\theta_1 y_i} (\cancel{g^{-\theta_1\Phi_i}} g^{\lambda_2 v_i t_{1,i}})^{-\beta\theta_2 y_i}\\
&= g'\cdot\prod_{i=1}^{n} K_{3,i}^{-f_{1,i}} K_{4,i}^{-f_{2,i}} K_{5,i}^{-\tilde{h}_{1,i}} K_{6,i}^{-\tilde{h}_{2,i}} g^{-\lambda_2 v_i\beta y_i(t_{2,i}\theta_1 - t_{1,i}\theta_2)}\\
&= g'\cdot\prod_{i=1}^{n} K_{3,i}^{-f_{1,i}} K_{4,i}^{-f_{2,i}} K_{5,i}^{-\tilde{h}_{1,i}} K_{6,i}^{-\tilde{h}_{2,i}} B^{-\tilde{\lambda}_2 v_i y_i\tilde{\Omega}} \Rightarrow K_A \text{ is computable.}
\end{aligned}$$

$$\begin{aligned}
K_B &= \prod_{i=1}^{n} g^{-(r_i+\Phi_i)} = \prod_{i=1}^{n} g^{-\tilde{r}_i + \frac{c_y\tilde{\lambda}_2 v_i x_i\beta}{\alpha} - \tilde{\Phi}_i - \frac{c_x\tilde{\lambda}_2 v_i y_i\beta}{\alpha}} = \prod_{i=1}^{n} g^{-(\tilde{r}_i+\tilde{\Phi}_i) + \frac{c_y\tilde{\lambda}_2\beta}{\alpha}(c_y v_i x_i - c_x v_i y_i)} =\\
&= g^{-\sum_{i=1}^{n}(\tilde{r}_i+\tilde{\Phi}_i)} g^{\frac{\tilde{\lambda}_2\beta}{\alpha}(c_x\sum_{i=1}^{n} v_i y_i - c_y\sum_{i=1}^{n} v_i x_i)} = \prod_{i=1}^{n} g^{-(\tilde{r}_i+\tilde{\Phi}_i)}\cdot g^{\frac{\tilde{\lambda}_2\beta}{\alpha}(c_y c_x - c_x c_y)} = \prod_{i=1}^{n} g^{-(\tilde{r}_i+\tilde{\Phi}_i)}
\end{aligned}$$

Finally, to compute K_B, notice that :

$$\sum_{i=1}^{n} r_i + \Phi_i = \sum_{i=1}^{n}\tilde{r}_i - \frac{c_y\tilde{\lambda}_2 v_i x_i\beta}{\alpha} + \frac{c_x\tilde{\lambda}_2 v_i y_i\beta}{\alpha} + \tilde{\Phi}_i = -\frac{c_x c_y\tilde{\lambda}_2\beta}{\alpha} + \frac{c_x c_y\tilde{\lambda}_2\beta}{\alpha} + \sum_{i=1}^{n}\tilde{r}_i + \tilde{\Phi}_i$$

$\Rightarrow K_B = \prod_{i=1}^{n} g^{-(\tilde{r}_i+\tilde{\Phi}_i)}$, So K_B is computable.

Generating the challenge ciphertext. The simulator generates the challenge ciphertext as in 4 except that, for the components $\mathsf{ct}_{5,i}, \mathsf{ct}_{6,i}, \mathsf{ct}'_{5,i}, \mathsf{ct}'_{6,i}$, the values $\{y_i\}_i$'s (rather than $\{x_i\}_i$'s) are used as power of Z:

$$\mathsf{ct}_{5,i} = T_{1,i}^{\tilde{s}_1} \cdot D^{\tilde{t}_{1,i}} \cdot H_{1,i}^{\tilde{s}_2} \cdot C^{\tilde{h}_{1,i}} \cdot Z^{\theta_1 y_i} \qquad \mathsf{ct}_{6,i} = T_{2,i}^{\tilde{s}_1} \cdot D^{\tilde{t}_{2,i}} \cdot H_{2,i}^{\tilde{s}_2} \cdot C^{\tilde{h}_{2,i}} \cdot Z^{\theta_2 y_i}$$
$$\mathsf{ct}'_{5,i} = T_{1,i}^{\tilde{s}'_1} \cdot D^{\tilde{t}_{1,i}} \cdot H_{1,i}^{\tilde{s}'_2} \cdot C^{\tilde{h}_{1,i}} \cdot Z^{\theta_1 y_i} \qquad \mathsf{ct}'_{6,i} = T_{2,i}^{\tilde{s}'_1} \cdot D^{\tilde{t}_{2,i}} \cdot H_{2,i}^{\tilde{s}'_2} \cdot C^{\tilde{h}_{2,i}} \cdot Z^{\theta_2 y_i}$$

Analysis of the game:

- $Z = g^{\beta(\eta+\tau)} \Rightarrow \mathsf{ct}_{5,i} = T_{1,i}^{s_1} H_{1,i}^{s_2} Z^{-\theta_1 y_i} Z^{\theta_1 y_i} = T_{1,i}^{s_1} H_{1,i}^{s_2}$: $\mathcal{A}$ interacts with H_3
- $Z = g^r \Rightarrow \mathsf{ct}_{5,i} = T_{1,i}^{s_1} H_{1,i}^{s_2} \cdot g^{-\beta(\eta+\tau)\theta_1 y_i} \cdot g^{\theta_1 y_i r} = T_{1,i}^{s_1} H_{1,i}^{s_2} \cdot g^{(r-\beta(\eta+\tau))\theta_1 y_i}$
  $\overset{s_4 = r-\beta(\eta+\tau)\neq 0}{\Rightarrow} \mathsf{ct}_{5,i} = T_{1,i}^{s_1} H_{1,i}^{s_3} V_1^{s_4 y_i}$: $\mathcal{A}$ interact with H_4 □

Proposition 6 *If the DLin assumption holds relative to* $\mathsf{GroupGen}$*, then* H_4 *and* H_5 *are computationally indistinguishable.*

Proof. The simulator takes as input $(g, A = g^{\alpha}, B = g^{\beta}, C = g^{\tau}, D = g^{\alpha\eta}, Z \overset{?}{=} g^{\beta(\eta+\tau)})$ and by interacting with the adversary $\mathcal{A}$, distinguish between the two cases $Z = g^{\beta(\eta+\tau)}$ and $Z \overset{\$}{\leftarrow} \mathbb{G}$, a random element of the group.

SetUp and token query phase. $\mathcal{B}$ runs as in the SetUp phase and token query phase in proposition 5.

Generating the challenge ciphertext. $\mathcal{B}$ chooses random elements $\tilde{s}_1, \tilde{s}_2, \tilde{s}_3, \tilde{s}_4, \tilde{s}'_1, \tilde{s}'_2, \tilde{s}'_3, k \leftarrow \mathbb{Z}_p^{\star}$ and computes the challenge ciphertext as follows:

- $\mathsf{ct}_1 = C \cdot g^{\tilde{s}_2} = g^{\tau+\tilde{s}_2} \Rightarrow s_2 = \tau + \tilde{s}_2$, $\bullet \mathsf{ct}'_1 = C^k \cdot g^{\tilde{s}'_2} = g^{k\tau+\tilde{s}'_2} \Rightarrow s'_2 = k\tau + \tilde{s}_2$
- $\mathsf{ct}_2 = D^{\tilde{\Omega}} \cdot A^{\tilde{\Omega}\tilde{s}_1} = (g^{\alpha\tilde{\Omega}})^{(\eta+\tilde{s}_1)} = h^{\eta+\tilde{s}_1} \Rightarrow s_1 = \eta + \tilde{s}_1$
- $\mathsf{ct}'_2 = D^{k\tilde{\Omega}} \cdot A^{\tilde{\Omega}\tilde{s}'_1} = (g^{\alpha\tilde{\Omega}})^{(k\eta+\tilde{s}'_1)} = h^{k\eta+\tilde{s}'_1} \Rightarrow s'_1 = k\eta + \tilde{s}'_1$
- $\mathsf{ct}_{3,i} = W_{1,i}^{\tilde{s}_1} \cdot F_{1,i}^{\tilde{s}_2} \cdot U_1^{\tilde{s}_3 x_i} \cdot D^{\tilde{w}_{1,i}} \cdot C^{f_{1,i}} = W_{1,i}^{\tilde{s}_1} \cdot F_{1,i}^{\tilde{s}_2+\tau} \cdot U_1^{\tilde{s}_3 x_i} \cdot g^{\eta\alpha\tilde{w}_{1,i}} \cdot F_{1,i}^{\tau} =$
  $= W_{1,i}^{\tilde{s}_1} \cdot F_{1,i}^{\tilde{s}_2+\tau} \cdot U_1^{\tilde{s}_3 x_i} \cdot g^{\eta(w_{1,i}-\beta\delta_1 x_i)} = W_{1,i}^{\tilde{s}_1+\eta} \cdot F_{1,i}^{\tilde{s}_2+\tau} \cdot U_1^{(\tilde{s}_3-\eta\beta)x_i} \Rightarrow s_3 = -\eta\beta + \tilde{s}_3$
- $\mathsf{ct}_{4,i} = W_{2,i}^{\tilde{s}_1} \cdot F_{2,i}^{\tilde{s}_2} \cdot U_2^{\tilde{s}_3 x_i} \cdot D^{\tilde{w}_{2,i}} \cdot C^{f_{2,i}}$, (similar computation as $\mathsf{ct}_{3,i}$)
- $\mathsf{ct}'_{3,i} = W_{1,i}^{\tilde{s}'_1} \cdot F_{1,i}^{\tilde{s}'_2} \cdot U_1^{\tilde{s}'_3 x_i} \cdot D^{k\tilde{w}_{1,i}} \cdot C^{kf_{1,i}} = W_{1,i}^{\tilde{s}'_1} \cdot F_{1,i}^{\tilde{s}'_2} \cdot U_1^{\tilde{s}'_3 x_i} \cdot g^{k\eta\alpha\tilde{w}_{1,i}} \cdot F_{1,i}^{k\tau}$
  $= W_{1,i}^{\tilde{s}'_1} \cdot F_{1,i}^{\tilde{s}'_2+k\tau} \cdot U_1^{\tilde{s}'_3 x_i} \cdot g^{k\eta(w_{1,i}-\beta\delta_1 x_i)} = W_{1,i}^{\tilde{s}'_1+k\eta} \cdot F_{1,i}^{\tilde{s}'_2+k\tau} \cdot U_1^{(\tilde{s}'_3-k\eta\beta)x_i} \Rightarrow s'_3 = -k\eta\beta + \tilde{s}'_3$
- $\mathsf{ct}'_{4,i} = W_{2,i}^{\tilde{s}'_1} \cdot F_{2,i}^{\tilde{s}'_2} \cdot U_2^{\tilde{s}'_3 x_i} \cdot D^{k\tilde{w}_{2,i}} \cdot C^{kf_{2,i}}$, (similar computation as $\mathsf{ct}'_{3,i}$)
- $\mathsf{ct}_{5,i} = T_{1,i}^{\tilde{s}_1} \cdot D^{\tilde{t}_{1,i}} \cdot H_{1,i}^{\tilde{s}_2} \cdot C^{\tilde{h}_{1,i}} \cdot Z^{\theta_1 y_i} \cdot g^{\tilde{s}_4\theta_1 y_i}$
- $\mathsf{ct}'_{5,i} = T_{1,i}^{\tilde{s}'_1} \cdot D^{k\tilde{t}_{1,i}} \cdot H_{1,i}^{\tilde{s}'_2} \cdot C^{k\tilde{h}_{1,i}} \cdot Z^{k\theta_1 y_i} \cdot g^{\tilde{s}_4\theta_1 y_i}$
- $\mathsf{ct}_{6,i} = T_{2,i}^{\tilde{s}_1} \cdot D^{\tilde{t}_{2,i}} \cdot H_{2,i}^{\tilde{s}_2} \cdot C^{\tilde{h}_{2,i}} \cdot Z^{\theta_2 y_i} \cdot g^{\tilde{s}_4\theta_2 y_i}$
- $\mathsf{ct}'_{6,i} = T_{2,i}^{\tilde{s}'_1} \cdot D^{k\tilde{t}_{2,i}} \cdot H_{2,i}^{\tilde{s}'_2} \cdot C^{k\tilde{h}_{2,i}} \cdot Z^{k\theta_2 y_i} \cdot g^{\tilde{s}_4\theta_2 y_i}$

Analysis of the game: First, notice that:

$$D^{\tilde{t}_{1,i}} = g^{\eta\alpha\tilde{t}_{1,i}} = g^{\eta(t_{1,i}-\beta\theta_1 y_i)} = T_{1,i}^{\eta} \cdot g^{-\beta\eta\theta_1 y_i}, D^{k\tilde{t}_{1,i}} = T_{1,i}^{k\eta} \cdot g^{-k\beta\eta\theta_1 y_i}$$

$$C^{\tilde{h}_{1,i}} = g^{\tau(h_{1,i}-\beta\theta_1 y_i)} = H_{1,i}^{\tau} \cdot g^{-\beta\tau\theta_1 y_i}, C^{k\tilde{h}_{1,i}} = H_{1,i}^{k\tau} \cdot g^{-k\beta\tau\theta_1 y_i} \Rightarrow$$

$$\begin{aligned}\mathsf{ct}_{5,i} &= T_{1,i}^{\tilde{s}_1} \cdot D^{\tilde{t}_{1,i}} \cdot H_{1,i}^{\tilde{s}_2} \cdot C^{\tilde{h}_{1,i}} \cdot (Z \cdot g^{\tilde{s}_4})^{\theta_1 y_i} = \\ &= T_{1,i}^{\eta+\tilde{s}_1} \cdot H_{1,i}^{\tau+\tilde{s}_2} \cdot (g^{-\beta(\tau+\eta)} \cdot Z \cdot g^{\tilde{s}_4})^{\theta_1 y_i} = T_{1,i}^{s_1} \cdot H_{1,i}^{s_2} \cdot (g^{(-\beta(\tau+\eta)} \cdot Z \cdot g^{\tilde{s}_4})^{\theta_1 y_i} \\ \mathsf{ct}'_{5,i} &= T_{1,i}^{s'_1} \cdot H_{1,i}^{s'_2} \cdot (g^{(-k\beta(\tau+\eta)} \cdot Z^k \cdot g^{\tilde{s}_4})^{\theta_1 y_i}\end{aligned}$$

$$\text{If } Z = g^{\beta(\eta+\tau)} \quad \Rightarrow \begin{cases} g^{-\beta(\tau+\eta)} \cdot Z \cdot g^{\tilde{s}_4} = g^{\tilde{s}_4} \Rightarrow \mathsf{ct}_{5,i} = T_{1,i}^{s_1} \cdot H_{1,i}^{s_2} \cdot U_1^{s_4 y_i} \\ g^{(-k\beta(\tau+\eta)} \cdot Z^k \cdot g^{\tilde{s}_4} = g^{\tilde{s}_4} \Rightarrow \mathsf{ct}_{5,i} = T_{1,i}^{s'_1} \cdot H_{1,i}^{s'_2} \cdot U_1^{s_4 y_i} \end{cases}$$

$$\Rightarrow \text{ The adversary interacts with hybrid } \mathsf{H}_4$$

$$\text{If } Z = g^{r} \quad \Rightarrow \begin{cases} g^{-\beta(\tau+\eta)} \cdot Z \cdot g^{\tilde{s}_4} = g^{r+\tilde{s}_4} \Rightarrow \mathsf{ct}_{5,i} = T_{1,i}^{s_1} \cdot H_{1,i}^{s_2} \cdot U_1^{s_4 y_i} \\ g^{(-k\beta(\tau+\eta)} \cdot Z^k \cdot g^{\tilde{s}_4} = g^{kr+\tilde{s}_4} \Rightarrow \mathsf{ct}_{5,i} = T_{1,i}^{s'_1} \cdot H_{1,i}^{s'_2} \cdot U_1^{s'_4 y_i} \end{cases}$$

$$\Rightarrow \text{ The adversary interacts with hybrid } \mathsf{H}_5$$

□

The proofs of indistinguishability for the other hybrids are similar to the ones that we have shown, that is $\mathsf{H}_5 - \mathsf{H}_6 \approx \mathsf{H}_4 - \mathsf{H}_5$, $\mathsf{H}_6 - \mathsf{H}_7 \approx \mathsf{H}_3 - \mathsf{H}_4$, $\mathsf{H}_7 - \mathsf{H}_8 \approx \mathsf{H}_2 - \mathsf{H}_3$, $\mathsf{H}_8 - \mathsf{H}_9 \approx \mathsf{H}_4 - \mathsf{H}_5$, $\mathsf{H}_9 - \mathsf{H}_{10} \approx \mathsf{H}_4 - \mathsf{H}_5$, $\mathsf{H}_{10} - \mathsf{H}_{11} \approx \mathsf{H}_1 - \mathsf{H}_2$ and $\mathsf{H}_{11} - \mathsf{H}_{12} \approx \mathsf{H}_0 - \mathsf{H}_1$.

4 Verifiable inner-product encryption

In this section we construct our (public-key) verifiable inner-product encryption.

Firstly, we present a formal definition of a VIPE scheme. Essentially, VIPE is similar to IPE except that it is endowed with extra verification algorithms $\mathsf{VrfyCT}, \mathsf{VrfyTok}$ and $\mathsf{VrfyMPK}$.

Definition 5. *A verifiable inner product encryption scheme for a message space $\mathcal{M}$ and for a family $\Sigma = \{\Sigma_n\}_{n>0}$ of vectors over some field is a tuple of PPT algorithms (here called VIP) with the following syntax and properties:*
$\mathsf{VIP} = \langle \mathsf{VIP.SetUp}, \mathsf{VIP.TokGen}, \mathsf{VIP.Enc}, \mathsf{VIP.Dec}, \mathsf{VIP.VrfyMPK}, \mathsf{VIP.VrfyCT}, \mathsf{VIP.VrfyTok} \rangle$

- $\mathsf{VIP.SetUp}(1^{\lambda}, n) \longrightarrow (\mathsf{MPK}, \mathsf{MSK})$*: as for IPE.*
- $\mathsf{VIP.TokGen}(\mathsf{MPK}, \mathsf{MSK}, \boldsymbol{v}) \longrightarrow \mathsf{Tok}_{\boldsymbol{v}}$*: as for IPE.*
- $\mathsf{VIP.Enc}(\mathsf{MPK}, \overrightarrow{x}, m) \longrightarrow \mathsf{CT}$*: as for IPE.*
- $\mathsf{VIP.Dec}(\mathsf{MPK}, \mathsf{Tok}_{\boldsymbol{v}}, \mathsf{CT}) \longrightarrow m \in \mathcal{M} \cup \{\perp\}$*: as for IPE.*
- $\mathsf{VIP.VrfyMPK}(\mathsf{MPK}) \longrightarrow \{0, 1\}$*: this is a deterministic algorithm that outputs* 1 *if* MPK *was correctly generated, or outputs* 0 *otherwise.*
- $\mathsf{VIP.VrfyCT}(\mathsf{MPK}, \mathsf{CT}) \longrightarrow \{0, 1\}$*: this is a deterministic algorithm that outputs* 1 *if* CT *was correctly generating using the master public key for some m in the message space $\mathcal{M}$, or outputs* 0 *otherwise.*

– $\mathsf{VIP.VrfyTok}(\mathsf{MPK}, \boldsymbol{v}, \mathsf{Tok}_{\boldsymbol{v}}) \longrightarrow \{0,1\}$*: this is a deterministic algorithm that outputs* 1 *if* $\mathsf{Tok}_{\boldsymbol{v}}$ *was correctly generated using the master secret key for vector* $\boldsymbol{v}$*, or outputs* 0 *otherwise.*
– Perfect correctness: *as for IPE.*
– Verifiability: VIP *is verifiable if for all* $\mathsf{MPK} \in \{0,1\}^*$*, all* $\mathsf{CT} \in \{0,1\}^*$*, there exists* $n > 0, (\boldsymbol{x}, m) \in \Sigma_n \times \mathcal{M}$ *such that for all* $\boldsymbol{v} \in \Sigma_n$ *and* $\mathsf{Tok}_{\boldsymbol{v}} \in \{0,1\}^*$*, the following holds:*

$$\begin{pmatrix} \mathsf{VIP.VrfyMPK}(\mathsf{MPK}) = 1 \wedge \\ \mathsf{VIP.VrfyCT}(\mathsf{MPK}, \mathsf{CT}) = 1 \wedge \\ \mathsf{VIP.VrfyTok}(\mathsf{MPK}, \boldsymbol{v}, \mathsf{Tok}_{\boldsymbol{v}}) = 1 \end{pmatrix} \Rightarrow \Pr\left[\, \mathsf{Dec}(\mathsf{MPK}, \boldsymbol{v}, \mathsf{Tok}_{\boldsymbol{v}}, \mathsf{CT}) = f_{\boldsymbol{v}}(m) \,\right] = 1$$

Intuitively verifiability states that each ciphertext (possibly with a maliciously generated public key) should be associated with a unique message $(\boldsymbol{x}, m)$ - up to parallelism - and decryption for a function $f_{\boldsymbol{v}}$ using any possibly maliciously generated token $\mathsf{Tok}_{\boldsymbol{v}}$ should result in $f_{\boldsymbol{v}}(x)$ for the unique message associated with the ciphertext [2].

4.1 Our construction

Our VIPE is based on a perfectly correct IPE (cf. our IPE scheme of Construction 1), a perfectly binding commitment scheme such as commitment scheme proposed in [10] and NIWI proofs for some specific relations that will be detailed below.

Let $n \in \mathbb{N}$ be the vector length and λ the security parameter, IP be a perfectly correct IPE scheme, Com be a perfectly binding commitment scheme and let $\mathsf{NIWI}^{\mathsf{mpk}} = \langle \mathcal{P}^{\mathsf{mpk}}, \mathcal{V}^{\mathsf{mpk}} \rangle$, $\mathsf{NIWI}^{\mathsf{enc}} = \langle \mathcal{P}^{\mathsf{enc}}, \mathcal{V}^{\mathsf{enc}} \rangle$ and $\mathsf{NIWI}^{\mathsf{tok}} = \langle \mathcal{P}^{\mathsf{tok}}, \mathcal{V}^{\mathsf{tok}} \rangle$ be NIWI proofs systems for, resp., the relations $\mathrm{R}^{\mathsf{mpk}}$, $\mathrm{R}^{\mathsf{enc}}$ and $\mathrm{R}^{\mathsf{tok}}$, that are essentially instantiation of analogous relations in [2]. The construction of these NIWI systems is provided in Section 5.

- $\mathrm{R}_{\mathsf{IP}}^{\mathsf{mpk}}(\overbrace{\mathsf{mpk}}^{x}, \overbrace{(\mathsf{msk}, \mathsf{r}^{\mathsf{mpk}})}^{w}) = \mathsf{TRUE} \iff (\mathsf{mpk}, \mathsf{msk}) = \mathsf{IP.SetUp}(1^\lambda, n; \mathsf{r}^{\mathsf{mpk}})$
- $\mathrm{R}_{\mathsf{IP}}^{\mathsf{tok}}\Big(\overbrace{(\mathsf{mpk}, t, \boldsymbol{v})}^{x}, \overbrace{(\mathsf{msk}, \mathsf{r}^{\mathsf{mpk}}, \mathsf{r}^{\mathsf{token}})}^{w}\Big) = \mathsf{TRUE} \iff \begin{pmatrix} (\mathsf{mpk}, (\mathsf{msk}, \mathsf{r}^{\mathsf{mpk}})) \in \mathrm{R}_{\mathsf{IP}}^{\mathsf{mpk}} \wedge \\ t = \mathsf{IP.TokGen}(\mathsf{MSK}, \boldsymbol{v}; \mathsf{r}^{\mathsf{tok}}) \end{pmatrix}$
- $\mathrm{R}_{\mathsf{IP}}^{k,\mathsf{ct}}\Big(\overbrace{\big((\mathsf{ct}_1, \mathsf{mpk}_1), \dots, (\mathsf{ct}_k, \mathsf{mpk}_k)\big)}^{x}, \overbrace{\big(\boldsymbol{x}, m, \mathsf{r}_1^{\mathsf{enc}}, \dots, \mathsf{r}_k^{\mathsf{enc}}\big)}^{w}\Big) = \mathsf{TRUE}, k \in [4]$
$\iff \forall i \in [k]\ \mathsf{ct}_i = \mathsf{IP.Enc}(\mathsf{mpk}_i, \boldsymbol{x}, m; \mathsf{r}_i^{\mathsf{enc}})$
- $\mathrm{R}^{\mathsf{enc}}(x, w) = \mathsf{TRUE} \iff \mathsf{P}_1^{\mathsf{enc}}(x, w) \vee \mathsf{P}_2^{\mathsf{enc}}(x, w)$, with
$\mathsf{P}_1^{\mathsf{enc}}\Big((\{c_i\}_{i\in[4]}, \{a_i\}_{i\in[4]}, z_0, z_1), (m, \boldsymbol{x}, \{\mathsf{r}_i^{\mathsf{enc}}\}_{i\in[4]}, i_1, i_2, \mathsf{r}_0^{\mathsf{com}}, \mathsf{r}_1^{\mathsf{com}})\Big) = \mathsf{TRUE}$
$\iff \Big(\big((c_1, a_1), \dots, (c_4, a_4)\big), (\boldsymbol{x}, m, \{\mathsf{r}_i^{\mathsf{enc}}\})\Big) \in \mathrm{R}_{\mathsf{IP}}^{4,\mathsf{ct}}$
$\mathsf{P}_2^{\mathsf{enc}}\Big((\{c_i\}_{i\in[4]}, \{a_i\}_{i\in[4]}, z_0, z_1), (m, \boldsymbol{x}, \{\mathsf{r}_i^{\mathsf{enc}}\}_{i\in[4]}, i_1, i_2, \mathsf{r}_0^{\mathsf{com}}, \mathsf{r}_1^{\mathsf{com}})\Big) = \mathsf{TRUE}$
$\iff \begin{pmatrix} i_1, i_2 \in [4] \wedge (i_1 \neq i_2) \wedge \Big(\big((c_{i_1}, a_{i_1}), (c_{i_2}, a_{i_2})\big), (\boldsymbol{x}, m, \mathsf{r}_i^{\mathsf{enc}})\Big) \in \mathrm{R}_{\mathsf{IP}}^{2,\mathsf{ct}} \\ \wedge\ z_0 = \mathsf{Com}(\{c_i\}_{i\in[4]}; \mathsf{r}_0^{\mathsf{com}}) \wedge\ z_1 = \mathsf{Com}(0; \mathsf{r}_1^{\mathsf{com}}) \end{pmatrix}$

- $\mathsf{R}^{\mathsf{tok}}(x,w) = \mathsf{TRUE} \iff \mathsf{P}_1^{\mathsf{tok}}(x,w) \vee \mathsf{P}_2^{\mathsf{tok}}(x,w)$, with, where

$$\mathsf{P}_1^{\mathsf{tok}}\Big((\boldsymbol{v}, \{t_i\}_{i\in[4]}, \{a_i\}_{i\in[4]}, z_0, z_1), (\{b_i\}_{i\in[4]}, \{\mathsf{r}_i^{\mathsf{mpk}}\}_{i\in[4]}, \{\mathsf{r}_i^{\mathsf{tok}}\}_{i\in[4]}, i_1, i_2, i_3, \mathsf{r}_0^{\mathsf{com}}, \mathsf{r}_1^{\mathsf{com}})\Big) = \mathsf{TRUE}$$

$$\iff \begin{pmatrix} \forall i \in [4] : \big((a_i, (b_i, \mathsf{r}_i^{\mathsf{mpk}})) \in \mathsf{R}^{\mathsf{mpk}} \wedge \ \big((a_i, t_i, \boldsymbol{v}_i), (b_i, \mathsf{r}_i^{\mathsf{mpk}}, \mathsf{r}_i^{\mathsf{tok}})\big)\big) \in \mathsf{R}_{\mathsf{IP}}^{\mathsf{tok}} \\ \wedge\ z_1 = \mathsf{Com}(1; \mathsf{r}_1^{\mathsf{com}}) \end{pmatrix}, \text{ and}$$

$$\mathsf{P}_2^{\mathsf{tok}}\big((\boldsymbol{v}, \{t_i\}_{i\in[4]}, \{a_i\}_{i\in[4]}, z_0, z_1), \Big(\{b_i\}_{i\in[4]}, \{\mathsf{r}_i^{\mathsf{mpk}}\}_{i\in[4]}, \{\mathsf{r}_i^{\mathsf{tok}}\}_{i\in[4]}, i_1, i_2, i_3, \mathsf{r}_0^{\mathsf{com}}, \mathsf{r}_1^{\mathsf{com}}\Big)\big) = \mathsf{TRUE}$$

$$\iff \begin{pmatrix} i_1, i_2, i_3 \in [4] \wedge \ (i_1 \neq i_2) \wedge (i_1 \neq i_3) \wedge (i_2 \neq i_3) \\ \forall j \in [3] : \big(a_{i_j}, (b_{i_j}, \mathsf{r}_{i_j}^{\mathsf{mpk}})\big) \in \mathsf{R}^{\mathsf{mpk}} \wedge \ \Big(\big(a_{i_j}, t_{i_j}, \boldsymbol{v}_{i_j}\big), \big(b_{i_j}, \mathsf{r}_{i_j}^{\mathsf{mpk}}, \mathsf{r}_{i_j}^{\mathsf{tok}}\big)\Big) \in \mathsf{R}_{\mathsf{IP}}^{\mathsf{tok}} \\ \wedge\ z_0 = \mathsf{Com}(\{c_i\}_{i\in[4]}; \mathsf{r}_1^{\mathsf{com}}) \wedge \ \exists m \in \mathcal{M}\ \forall i \in [4]\ \mathsf{IP.Dec}(c_i, t_i) = f_{\boldsymbol{v}}(m) \end{pmatrix}$$

Construction 7 [Our VIPE, VIP]

- $\mathsf{VIP.SetUp}(1^\lambda, n) \longrightarrow (\mathsf{MPK}, \mathsf{MSK})$:
  1. For $i \in [4]$, run $\mathsf{IP.SetUp}(1^\lambda, n)$ to generate $(\mathsf{MPK}_i, \mathsf{MSK}_i)$ as output.
  2. Run the commitment algorithm to generate $\mathsf{Z}_0 = \mathsf{Com}(0; \mathsf{r}_0^{\mathsf{com}})$ and $\mathsf{Z}_1 = \mathsf{Com}(1; \mathsf{r}_1^{\mathsf{com}})$.
  3. Output $\mathsf{VIP.MPK} = (\{\mathsf{MPK}_i\}_{i\in[4]}, \mathsf{Z}_0, \mathsf{Z}_1), \mathsf{VIP.MSK} = (\{\mathsf{MSK}_i\}_{i\in[4]}, u_0, u_1)$.
- $\mathsf{VIP.Enc}(\mathsf{MPK}, m, \boldsymbol{x}) \longrightarrow \mathsf{CT}$:
  1. For $i \in [4]$, run the encryption algorithm to compute $\mathsf{CT}_i = \mathsf{IP.Enc}(\mathsf{MPK}, m, \boldsymbol{x}; \mathsf{r}_i^{\mathsf{enc}})$.
  2. Set $x = (\{\mathsf{CT}_i\}_{i\in[4]}, \{\mathsf{MPK}_i\}_{i\in[4]}, \mathsf{Z}_0, \mathsf{Z}_1), w = (m, \boldsymbol{x}, \{\mathsf{r}_i^{\mathsf{enc}}\}_{i\in[4]}, 0, 0, 0^{|u_0|}, 0^{|u_1|})$, and run $\mathcal{P}^{\mathsf{enc}}(x,w)$ to generate π_{ct} for relation $\mathsf{R}^{\mathsf{enc}}(x,w)$. Note that $\mathsf{P}_1^{\mathsf{enc}}(x,w) = \mathsf{TRUE}$
  3. Output ciphertext $\mathsf{CT} = (\{\mathsf{CT}_i\}_{i\in[4]}, \pi_{\mathsf{ct}})$.
- $\mathsf{VIP.TokGen}(\mathsf{MPK}, \mathsf{MSK}, f_{\boldsymbol{v}})$:
  1. For $i \in [4]$, run $\mathsf{IP.TokGen}(\mathsf{MSK}, \boldsymbol{v}; \mathsf{r}_i^{\mathsf{tok}})$ to generate $\mathsf{Tok}_{\boldsymbol{v}}^i$.
  2. $x = (\boldsymbol{v}, \{\mathsf{Tok}_{\boldsymbol{v}}^i\}_{i\in[4]}, \{\mathsf{MPK}_i\}_{i\in[4]}, \mathsf{Z}_0, \mathsf{Z}_1), w = (\{\mathsf{MSK}_i\}_{i\in[4]}, \{\mathsf{r}_i^{\mathsf{tok}}\}_{i\in[4]}, 0, 0, 0, 0^{|u_0|}, u_1)$ run $\mathcal{P}^{\mathsf{tok}}$ to generate π_{tok} to prove $\mathsf{R}^{\mathsf{tok}}(x,w) = \mathsf{TRUE}$. Note that $\mathsf{P}_1^{\mathsf{tok}}(x,w) = \mathsf{TRUE}$
  3. Output token $\mathsf{Tok}_{\boldsymbol{v}} = (\{\mathsf{Tok}_{\boldsymbol{v}}^i\}_{i\in[4]}, \pi_{\mathsf{tok}})$.
- $\mathsf{VIP.Dec}(\mathsf{MPK}, f_{\boldsymbol{v}}, \mathsf{Tok}_{\boldsymbol{v}}, \mathsf{CT})$:
  1. Run the verification algorithms $\mathcal{V}^{\mathsf{mpk}}, \mathcal{V}^{\mathsf{enc}}, \mathcal{V}^{\mathsf{tok}}$ with inputs the corresponding pairs of statement and proof (the proof for the verification of the master public key is set to the empty string). If some verification algorithms fails, then stop and output $\perp$ or go to the next step otherwise.
  2. For all $i \in [4]$, compute $m^{(i)} = \mathsf{IP.Dec}(\mathsf{Tok}_{\boldsymbol{v}}^{(i)}, \mathsf{CT}_i)$ and output the following:
  $$\begin{cases} \texttt{If } \exists i_1, i_2, i_3 \in [4] \ \ s.t.\ \ m = m^{(i_1)} = m^{(i_2)} = m^{(i_3)} \Rightarrow \texttt{Output } m. \\ \texttt{If } \nexists i_1, i_2, i_3 \in [4] \ \ s.t.\ \ m^{(i_1)} = m^{(i_2)} = m^{(i_3)} \Rightarrow \texttt{Output } \perp. \end{cases}$$
- $\mathsf{VIP.VrfyMPK}(\mathsf{MPK})$: run $\mathcal{V}^{\mathsf{mpk}}(\mathsf{MPK}, \epsilon)$ and output its result.
- $\mathsf{VIP.VrfyCT}\big((\{\mathsf{CT}_i\}_{i\in[4]}, \{\mathsf{MPK}_i\}_{i\in[4]}, \mathsf{Z}_0, \mathsf{Z}_1), \pi_{\mathsf{ct}})\big)$:
  run $\mathcal{V}^{\mathsf{enc}}\big((\{\mathsf{CT}_i\}_{i\in[4]}, \{\mathsf{MPK}_i\}_{i\in[4]}, \mathsf{Z}_0, \mathsf{Z}_1), \pi_{\mathsf{ct}}\big)$ and output its result.
- $\mathsf{VIP.VrfyTok}\big((\boldsymbol{v}, \{\mathsf{Tok}_{\boldsymbol{v}}^i\}_{i\in[4]}, \{\mathsf{MPK}_i\}_{i\in[4]}, \mathsf{Z}_0, \mathsf{Z}_1), \pi_{\mathsf{tok}}\big)$:
  run $\mathcal{V}^{\mathsf{tok}}\big((\boldsymbol{v}, \{\mathsf{Tok}_{\boldsymbol{v}}^i\}_{i\in[4]}, \{\mathsf{MPK}_i\}_{i\in[4]}, \mathsf{Z}_0, \mathsf{Z}_1), \pi_{\mathsf{tok}}\big)$ and output its result.

Correctness of VIP follows from perfect correctness of IP. IND-Security and Verifiability of VIP follows as corollary (following theorem 2) from the verifiability and IND-Security of the construction of [2] for general functions.

Theorem 2. *If* IP *is a perfectly correct* IND*-Secure IP scheme for message space* $\mathcal{M}$ *and for the set* $\mathbb{Z}_p^n$ *of vectors of length* n *over* $\mathbb{Z}_p$*, and* $\mathsf{NIWI}^{\mathsf{mpk}}, \mathsf{NIWI}^{\mathsf{ct}}, \mathsf{NIWI}^{\mathsf{tok}}$ *are NIWI systems resp. for the relations* $R^{\mathsf{mpk}}, R^{\mathsf{enc}}, R^{\mathsf{tok}}$ *and* Com *is a non-interactive perfectly binding and computationally hiding commitment scheme, then* VIP *is an* IND*-Secure VIPE scheme for the class of inner product functionality over* $\mathcal{M}$ *and* $\mathbb{Z}_p^n$*.*

5 NIWI Proofs and Verification algorithms

In this section we present the proof systems that we used in our VIP scheme, to prove membership of relations R^{mpk}, R^{tok} and R^{enc}. For each of our relations[2], we need to define a system of equations such that satisfiability of that system and the membership in the relation are equivalent. Then, the GS generic prover and verifier algorithms, $\mathsf{NIWI}_{\mathsf{GS}} = \langle \mathcal{P}_{\mathsf{GS}}, \mathcal{V}_{\mathsf{GS}} \rangle$, can be used for such equations. In this section, for each of our relations of Section 4, we will either define a corresponding system of equations or we will show how to implement directly (without using GS proofs).

Definition 6 (Pairing Product System of Equations). *Consider bilinear map* $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$*. The following system of equation with* k *equations over* m *variables* $\mathcal{X}_i \in \mathbb{G}, i \in [m]$ *and constants* $B_i^{(t)} \in \mathbb{G}, \tau^{(t)} \in \mathbb{G}_T$ *and* $\gamma_{ij}^{(t)} \in \mathbb{Z}_p$ *for* $i \in [m], t \in [k]$ *is called a* pairing product system of equations *over* $(\mathbb{G}, \mathbb{G}_T, e)$*:*

$$\mathsf{E} : \begin{cases} \prod_{i=1}^m \mathbf{e}(\mathcal{X}_i, B_i^{(1)}) \cdot \prod_{i=1}^m \prod_{j=1}^m \mathbf{e}(\mathcal{X}_i, \mathcal{X}_j)^{\gamma_{ij}^{(1)}} = \tau^{(1)} \\ \dots \\ \prod_{i=1}^m \mathbf{e}(\mathcal{X}_i, B_i^{(k)}) \cdot \prod_{i=1}^m \prod_{j=1}^m \mathbf{e}(\mathcal{X}_i, \mathcal{X}_j)^{\gamma_{ij}^{(k)}} = \tau^{(k)} \end{cases} \tag{1}$$

$(g_1, g_2, \dots, g_m) \in \mathbb{G}^m$ *is a solution for the equation* E *iff*

$$\Big(\mathsf{E}[(g_1, \dots, g_m)] = \mathsf{TRUE}\Big) = \begin{cases} \prod_{i=1}^m \mathbf{e}(g_i, B_i^{(1)}) \cdot \prod_{i=1}^m \prod_{j=1}^m \mathbf{e}(g_i, g_j)^{\gamma_{ij}^{(1)}} = \tau^{(1)} \\ \dots \\ \prod_{i=1}^m \mathbf{e}(g_i, B_i^{(k)}) \cdot \prod_{i=1}^m \prod_{j=1}^m \mathbf{e}(g_i, g_j)^{\gamma_{ij}^{(k)}} = \tau^{(k)} \end{cases}$$

We define the following relation for pairing product system of equations:

$$R_{\boldsymbol{E}} = \{(x, w) |\ x = \mathsf{E}, w = (g_1, \dots, g_m) : \mathsf{E}[(g_1, \dots, g_m)] = \mathsf{TRUE}\}$$

Throughout the paper, we denote by $\mathsf{NIWI}_{\mathsf{GS}} = \langle \mathcal{P}_{\mathsf{GS}}, \mathcal{V}_{\mathsf{GS}} \rangle$ a Groth-Sahai [11] NIWI-proof system. Precisely:

- $\mathcal{P}_{\mathsf{GS}}(x = \mathsf{E}, w = (g_1, \dots, g_m)) \to \pi_{\mathsf{E}}$ • $\mathcal{V}_{\mathsf{GS}}(x, \pi_{\mathsf{E}}) \to \begin{cases} 1 : & \texttt{If } (x, w) \in R_{\mathsf{E}} \\ 0 : & \texttt{Otherwise} \end{cases}$

[2] Actually, we will implement some or part of them not directly using GS proofs.

5.1 How to handle generalized OR statements

Some of our relations of Section 4 consist of a generalized form of disjunction (OR) of two predicates, let us say P_1 and P_2. Suppose that we have equivalent systems of equations for each of the two predicate, that is a system of equations E_1 (resp. E_2) representing predicate P_1 (resp. P_2). Consider the following relation:

$$\mathcal{R}_{\mathsf{OR}} = \{(x, w) |\ x = (\mathsf{E}_1, \mathsf{E}_2), w = (\mathsf{idx}, w_1, w_2) : \mathsf{idx} \in \{1, 2\}\ \wedge\ (\mathsf{E}_{\mathsf{idx}}, w_{\mathsf{idx}}) \in \mathcal{R}_{\mathsf{E}}\ \wedge\ w_{\bar{\mathsf{idx}}} \in \mathbb{G}^3\},$$

where $\bar{\mathsf{idx}}$ means $\{1, 2\} - \{\mathsf{idx}\}$.

Notice that the relation is not exactly a disjunction of pairing product equations because we need to make sure that the statement that holds is the one selected by the index in the witness, so we cannot use the technique of Groth [9] and we will follow a different approach.

By hypothesis $\mathcal{P}_{\mathsf{GS}}$ takes as input a system of equations E as statement and a solution $(g_1, \ldots, g_m)$ as witness and provides a NIWI-proof of membership of $(\mathsf{E}, w) \in \mathcal{R}_{\mathsf{E}}$. Therefore, to use $\mathsf{NIWI}_{\mathsf{GS}}$ to generate a NIWI-proof for relation $\mathcal{R}_{\mathsf{OR}}$, we need to define a third system of equation E_{OR} with the following properties:

1. $\mathsf{E}_{\mathsf{OR}} \approx \mathcal{R}_{\mathsf{OR}}$. With this notation, we mean that there exist two efficiently computable functions f and g such that:

$$\exists w = (\mathsf{idx}, w_1, w_2)\ \big(x = (\mathsf{E}_1, \mathsf{E}_2), w\big) \in \mathcal{R}_{\mathsf{OR}} \Leftrightarrow \exists \tilde{w}\ \big(\mathsf{E}_{\mathsf{OR}} = f(x), \tilde{w}\big) \in \mathcal{R}_{\mathsf{E}}.$$

$$\big(x, w\big) \in \mathcal{R}_{\mathsf{OR}} \Rightarrow \big(f(x), g(x, w)\big) \in \mathcal{R}_{\mathsf{OR}}.$$

   The latter properties guarantee that a proof for relation $\mathcal{R}_{\mathsf{OR}}$ computed using $\mathsf{NIWI}_{\mathsf{GS}}$ satisfies completeness and soundness. For WI to hold, we need the following property.
2. The function f is efficiently invertible.

Now we show how to construct the system of equations E_{OR} with the aforementioned properties.

Consider two systems of pairing product equations E_1 and E_2 - same structure as in 1. For simplicity, we assume the equations are over two variables (the general case is straightforward).

$$\mathsf{E}_1 : \mathbf{e}(\mathcal{X}_1, a_1) \cdot \mathbf{e}(\mathcal{X}_2, a_2) = \tau_1\ , \mathsf{E}_2 : \mathbf{e}(\mathcal{Y}_1, b_1) \cdot \mathbf{e}(\mathcal{Y}_2, b_2) = \tau_2$$

We define the new system of equation E_{OR} with 4 new variables $\mathcal{Z}_{11}, \mathcal{Z}_{12}, \mathcal{Z}_{21}, \mathcal{Z}_{22}$ as follows:

$$\mathsf{E}_{\mathsf{OR}} : \begin{cases} \mathbf{e}(\mathcal{X}_1, a_1) \cdot \mathbf{e}(\mathcal{X}_2, a_2) \cdot \mathbf{e}(\mathcal{Z}_{11}, \mathcal{Z}_{12}) = \tau_1 \\ \mathbf{e}(\mathcal{Y}_1, b_1) \cdot \mathbf{e}(\mathcal{Y}_2, b_2) \cdot \mathbf{e}(\mathcal{Z}_{21}, \mathcal{Z}_{22}) = \tau_2 \\ \mathbf{e}(\mathcal{Z}_{11}, \mathcal{Z}_{22}) = 1 \\ \mathbf{e}(\mathcal{Z}_{11}, g) \cdot \mathbf{e}(\mathcal{Z}_{\mathsf{idx}}, g) = \mathbf{e}(g, g) \\ \mathbf{e}(\mathcal{Z}_{22}, g) \cdot \mathbf{e}(\mathcal{Z}_{\mathsf{idx}}, g) = \mathbf{e}(g^2, g) \end{cases}$$

Analysis of the equations: Consider $(\mathcal{Z}_{\mathsf{idx}} \hookleftarrow g_{\mathsf{idx}}, \mathcal{X}_1 \hookleftarrow g_1, \mathcal{X}_2 \hookleftarrow g_2, \mathcal{Y}_1 \hookleftarrow g_3, \mathcal{Y}_2 \hookleftarrow g_4, \mathcal{Z}_{11} \hookleftarrow g_{11}, \ldots, \mathcal{Z}_{22} \hookleftarrow g_{22})$ as a solution for E_{OR}. So, there exist values $\mathsf{idx}, z_{11}, z_{22} \in \mathbb{Z}_p$ such that $g_{\mathsf{idx}} = g^{\mathsf{idx}}, g_{11} = g^{z_{11}}, g_{22} = g^{z_{22}}$ and for $t \in [k]$ there exist values α_t such that $\tau_t = \mathbf{e}(g, \alpha_t)$.

- $\mathbf{e}(\mathcal{Z}_{11}, g) \cdot \mathbf{e}(\mathcal{Z}_{\mathsf{idx}}, g) = \mathbf{e}(g, g) \Rightarrow \mathbf{e}(g^{z_{11}+\mathsf{idx}-1}, g) = 1$
  $\Rightarrow z_{11} = 1 - \mathsf{idx}$ and similarly $z_{22} = 2 - \mathsf{idx}$.
- $\mathbf{e}(\mathcal{Z}_{11}, \mathcal{Z}_{22}) = 1 \Rightarrow (z_{11} = 0 \ \vee \ z_{22} = 0)$
- $z_{11} = 0 \ \wedge \ z_{11} = 1 - \mathsf{idx} \Rightarrow \mathbf{e}(\mathcal{X}_1 \hookleftarrow g_1, a_1) \cdot \mathbf{e}(\mathcal{X}_2 \hookleftarrow g_2, a_2) = \tau_1$
  $\Rightarrow (\mathsf{E}_1[g_1, g_2] = \mathsf{TRUE} \ \wedge \ \mathsf{idx} = 1)$
- Similarly, $z_{22} = 0 \ \wedge z_{22} = 2 - \mathsf{idx} \Rightarrow \mathbf{e}(\mathcal{Z}_{21}, \mathcal{Z}_{22}) = 1 \Rightarrow (\mathsf{E}_2[g_3, g_4] = \mathsf{TRUE} \ \wedge \ \mathsf{idx} = 2)$

The above facts imply that:

$$\mathsf{E}_{\mathsf{OR}}[(g_{\mathsf{idx}}, g_1, \ldots, g_4, g_{11}, \ldots, g_{22})] = \mathsf{TRUE} \Rightarrow$$
$$\Big((\mathsf{E}_1[g_1, g_2, \alpha_1] = \mathsf{TRUE} \ \wedge \ \mathsf{idx} = 1\Big) \vee \Big(\mathsf{E}_2[g_3, g_4, \alpha_2] = \mathsf{TRUE} \ \wedge \ \mathsf{idx} = 2)\Big),$$

as it was to show. It is also easy to see that the previous transformation is efficiently invertible.

For the other direction, suppose w.l.o.g that $w_1 = (g_1, g_2, \alpha_1)$ is a solution to $x = \mathsf{E}_1$ (the other case is symmetrical and we omit it), namely $(x, w_1) \in \mathcal{R}'$. Suppose also that $w_2 = (g_3, g_4, \alpha_2) \in \mathbb{G}^3$ is an arbitrary triple of elements of $\mathbb{G}$. Therefore $(1, w_1, w_2)$ is a witness to $(\mathsf{E}_1, \mathsf{E}_2)$ with respect to relation $\mathcal{R}_{\mathsf{OR}}$. Then, setting $(\mathcal{Z}_{\mathsf{idx}} \hookleftarrow g^1, \mathcal{X}_1 \hookleftarrow g_1, \mathcal{X}_2 \hookleftarrow g_2, \mathcal{Y}_1 \hookleftarrow g^0, \mathcal{Y}_2 \hookleftarrow g^0, \mathcal{Z}_{11} \hookleftarrow g^0, \mathcal{Z}_{12} \hookleftarrow g^1, \mathcal{Z}_{21} \hookleftarrow \alpha_2, \mathcal{Z}_{22} \hookleftarrow g^1)$, we have that:

$$\mathsf{E}_{\mathsf{OR}}[(g_{\mathsf{idx}}, g_1, \ldots, g_4, g_{11}, \ldots, g_{22})] = \mathsf{TRUE}.$$

(Notice that we implicitly defined a transformation g as needed.)

5.2 OR proof in the general case

If the number of pairing products (m) in each of the two equations is greater than 1, such as:

$$\mathsf{E}_1 : \begin{cases} \mathbf{e}(\mathcal{X}_1, a_1) \cdot \mathbf{e}(\mathcal{X}_2, a_2) = \tau_1 \\ \mathbf{e}(\mathcal{X}_1, a'_1) \cdot \mathbf{e}(\mathcal{X}_2, a'_2) = \tau'_1 \end{cases}, \quad \mathsf{E}_2 : \begin{cases} \mathbf{e}(\mathcal{Y}_1, b_1) \cdot \mathbf{e}(\mathcal{Y}_2, b_2) = \tau_2 \\ \mathbf{e}(\mathcal{Y}_1, b'_1) \cdot \mathbf{e}(\mathcal{Y}_2, a'_2) = \tau'_2 \end{cases}$$

then E_{OR} can be defined as:

$$\mathsf{E}_{\mathsf{OR}} : \begin{cases} \mathbf{e}(\mathcal{X}_1, a_1) \cdot \mathbf{e}(\mathcal{X}_1, a_2) \cdot \mathbf{e}(\mathcal{Z}_{11}, \mathcal{Z}_{12}) = \tau_1 \\ \mathbf{e}(\mathcal{X}_1, a'_1) \cdot \mathbf{e}(\mathcal{X}_2, a'_2) \cdot \mathbf{e}(\mathcal{Z}_{11}, \mathcal{Z}_{13}) = \tau'_1 \\ \mathbf{e}(\mathcal{Y}_1, b_1) \cdot \mathbf{e}(\mathcal{Y}_2, b_2) \cdot \mathbf{e}(\mathcal{Z}_{21}, \mathcal{Z}_{22}) = \tau_2 \\ \mathbf{e}(\mathcal{Y}_1, b'_1) \cdot \mathbf{e}(\mathcal{Y}_2, b'_2) \cdot \mathbf{e}(\mathcal{Z}_{23}, \mathcal{Z}_{22}) = \tau'_2 \\ \mathbf{e}(\mathcal{Z}_{11}, \mathcal{Z}_{22}) = 1 \\ \mathbf{e}(\mathcal{Z}_{11}, g) \cdot \mathbf{e}(\mathcal{Z}_{\mathsf{idx}}, g) = \mathbf{e}(g, g) \\ \mathbf{e}(\mathcal{Z}_{22}, g) \cdot \mathbf{e}(\mathcal{Z}_{\mathsf{idx}}, g) = \mathbf{e}(g^2, g) \end{cases}$$

We omit further details.

Notations: For the rest of this section, let us fix $n \in \mathbb{N}$ as dimension of the vector space and let $i \in [n], b \in [2]$. Note we can efficiently check whether a string is a valid group element. We recall what follows.

$$\mathsf{mpk} = (g, h, \{W_{b,i}, F_{b,i}, T_{b,i}, H_{b,i}, U_b, V_b\}, K_1, K_2, \Lambda) \in \mathbb{G}^{4n+8} \times \mathbb{G}_T$$
$$\mathsf{msk} = (\{w_{b,i}, f_{b,i}, t_{b,i}, h_{b,i}, \delta_b, \theta_b\}, \Omega, k) \in \mathbb{Z}_p^{4n+6}$$
$$\mathsf{tok} = (K_A, K_B, \{K_{3,i}, K_{4,i}, K_{5,i}, K_{6,i}\}_i) \in \mathbb{G}^{4n+2}$$
$$\mathsf{ct} = \Big((\mathsf{ct}_1, \mathsf{ct}_2, \begin{Bmatrix} \mathsf{ct}_{3,i} \ , \ \mathsf{ct}_{4,i} \\ \mathsf{ct}_{5,i} \ , \ \mathsf{ct}_{6,i} \end{Bmatrix}, \mathsf{ct}_7, \mathsf{ct}_8), (\mathsf{ct}'_1, \mathsf{ct}'_2, \begin{Bmatrix} \mathsf{ct}'_{3,i} \ , \ \mathsf{ct}'_{4,i} \\ \mathsf{ct}'_{5,i} \ , \ \mathsf{ct}'_{6,i} \end{Bmatrix}, \mathsf{ct}'_7, \mathsf{ct}'_8)\Big) \in \mathbb{G}^{8n+6} \times \mathbb{G}_T^2$$

5.3 Master Public Key Verification

Let $x = \mathsf{mpk}$. Since g and $\mathbf{e}(g, g)$ are generators for the groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p, we can represent all components of x as a power of either g or $\mathbf{e}(g, g)$. That is, there exist $\Omega, k', \{w_{b,i}, f_{b,i}, t_{b,i}, h_{b,i}\}, \{\delta_b, \theta_b, k_b\}$ for $i \in [n]$ and $b \in [2]$, in $\mathbb{Z}_p$ such that: $h = g^{\Omega}, \Lambda = \mathbf{e}(g, g)^{k'}, W_{b,i} = g^{w_{b,i}}, F_{b,i} = g^{f_{b,i}}, T_{b,i} = g^{t_{b,i}}, H_{b,i} = g^{h_{b,i}}, U_b = g^{\delta_b}, V_b = g^{\theta_b}, K_b = g^{k_b}$. The following holds:

$$\begin{aligned} \mathbf{e}(g, h) =& \mathbf{e}(U_1, W_{2,i}) \cdot \mathbf{e}(U_2, W_{1,i})^{-1} \Rightarrow \mathbf{e}(g, g^{\Omega}) = \mathbf{e}(g^{\delta_1}, g^{w_{2,i}}) \cdot \mathbf{e}(g^{\delta_2}, g^{-w_{1,i}}) \\ & \Rightarrow \Omega = \delta_1 w_{2,i} - \delta_2 w_{1,i}. \\ \mathbf{e}(g, h) =& \mathbf{e}(V_1, T_{2,i}) \cdot \mathbf{e}(V_2, T_{1,i})^{-1} \Rightarrow \mathbf{e}(g, g^{\Omega}) = \mathbf{e}(g^{\theta_1}, g^{t_{2,i}}) \cdot \mathbf{e}(g^{\theta_2}, g^{-t_{1,i}}) \\ & \Rightarrow \Omega = \theta_1 t_{2,i} - \theta_2 t_{1,i}. \end{aligned}$$

$$\mathbf{e}(K_1, K_2) = \mathbf{e}(g^{k_1}, g^{k_2}) = \Lambda = \mathbf{e}(g, g^{k'}) \Rightarrow k' = k_1 k_2$$

By defining $g' = g^{k'}, K_1 = g^{k_1}, K_2 = g^{k_2}$, it follows that: $\Lambda = e(K_1, K_2), K_1 = g^k, K_2 = g'^{\frac{1}{k}}$

Hence, we have the following verification algorithm for master public key:

```
Input: mpk, Output: 1 if mpk is a well-generated master public key for IP scheme and 0 otherwise
(1) If $\Lambda \neq \mathbf{e}(K_1, K_2)$. output 0 otherwise go to the next step
(2) For $i = 1$to $n$ do :
        $(i.a)$ If $\mathbf{e}(U_1, W_{2,i}) \cdot \mathbf{e}(U_2, W_{1,i})^{-1} \neq \mathbf{e}(h, g)$, output 0 otherwise go to the next step
        $(i.b)$ If $\mathbf{e}(V_1, T_{2,i}) \cdot \mathbf{e}(V_2, T_{1,i})^{-1} \neq \mathbf{e}(h, g)$, output 0 otherwise go to the next step
(3) Output 1.
```

Fig. 2. Master public key verification algorithm. (membership in relation $\mathrm{R}_{\mathsf{IP}}^{\mathsf{mpk}}$)

5.4 Token verification algorithms

As it was defined in section, there are two relation for token, $Ŗ^{\mathsf{tok}}_{\mathsf{IP}}$ and $Ŗ^{\mathsf{tok}}$. The following algorithm verifies membership in relation $Ŗ^{\mathsf{tok}}_{\mathsf{IP}}$.

```
Input: $\mathsf{MPK}, \boldsymbol{v} = (v_1, \dots, v_n) \neq \mathbf{0}, \mathsf{tok}$
Output: 1 if $\mathsf{tok}$ is a well-generated token for $\mathsf{IP}$ scheme and 0 otherwise

1. If $\boldsymbol{v} = \mathbf{0}$ output 0 else put index $i^*$ such that $v_{i^*} \neq 0$
2. Compute $\Lambda_1^* = \mathbf{e}(K_{3,i}, U_1) \cdot \mathbf{e}(K_{4,i}, U_2)$ and $\Lambda_2^* = \mathbf{e}(K_{5,i}, V_1) \cdot \mathbf{e}(K_{6,i}, V_2)$
3. If $\Lambda_1^* = 1_{\mathbb{G}_T}$ OR $\Lambda_2^* = 1_{\mathbb{G}_T}$ output $\perp$
4. For $i = 1$ to $n$ do:
   (a) If $\Big(\mathbf{e}(K_{3,i}, U_1) \cdot \mathbf{e}(K_{4,i}, U_2)\Big)^{v_{i^*}} \neq (\Lambda_1^*)^{v_i}$ output 0
   (b) If $\Big(\mathbf{e}(K_{5,i}, V_1) \cdot \mathbf{e}(K_{6,i}, V_2)\Big)^{v_{i^*}} \neq (\Lambda_2^*)^{v_i}$ output 0
5. If $\mathbf{e}(K_A, g) \neq \Lambda \prod_{i=1}^n \mathbf{e}(K_{3,i}, F_{1,i})^{-1}.\mathbf{e}(K_{4,i}, F_{2,i})^{-1}.\mathbf{e}(K_{5,i}, H_{1,i})^{-1}\mathbf{e}(K_{6,i}, H_{2,i})^{-1}$ output 0
6. If $\prod_{i=1}^n \mathbf{e}(K_{3,i}, W_{1,i}) \cdot \mathbf{e}(K_{4,i}, W_{2,i}) \cdot \mathbf{e}(K_{5,i}, T_{1,i}) \cdot \mathbf{e}(K_{6,i}, T_{2,i}) \neq \mathbf{e}(h, K_B)^{-1}$ output 0
7. output 1.
```

Fig. 3. First token verification algorithm. (membership in relation $Ŗ^{\mathsf{tok}}_{\mathsf{IP}}$)

Correctness of the algorithm: For simplicity let's assume $v_1 \neq 0$ and $i^* = 1$.

$$\bullet \Lambda_1^*, \Lambda_2^* \in \mathbb{G}_T \Rightarrow \exists \lambda_1, \lambda_2 \in \mathbb{Z}_p \;\; s.t. \;\; \Lambda_1^* = e(g,h)^{\lambda_1 v_1}, \Lambda_2^* = e(g,h)^{\lambda_2 v_1}$$

$$\bullet \forall i \in [n] \; \exists r_i, r_i' \in \mathbb{Z}_p \;\; s.t. \;\; K_{3,i} = g^{-\delta_2 r_i} \cdot g^{\lambda_1 v_i w_{2,i}}, K_{4,i} = g^{\delta_1 r_i'} \cdot g^{-\lambda_1 v_i w_{1,i}}$$

$$\Rightarrow \mathbf{e}(K_{3,i}, U_1) \cdot \mathbf{e}(K_{4,i}, U_2) = \mathbf{e}(g^{-\delta_2 r_i} \cdot g^{\lambda_1 v_i w_{2,i}}, g^{\delta_1}) \cdot \mathbf{e}(g^{\delta_1 r_i'} \cdot g^{-\lambda_1 v_i w_{1,i}}, g^{\delta_2}) =$$
$$\mathbf{e}(g,g)^{\delta_1 \delta_2 (r_i' - r_i)} \cdot \mathbf{e}(g,h)^{\lambda_1 v_i} =$$
$$\Rightarrow \Big(\mathbf{e}(K_{3,i}, U_1) \cdot \mathbf{e}(K_{4,i}, U_2)\Big)^{v_1} = \mathbf{e}(g,g)^{v_1 \delta_1 \delta_2 (r_i' - r_i)} \cdot \mathbf{e}(g,h)^{\lambda_1 v_1 v_i}$$

- **Step 3:** $\Lambda_1^* \neq 1_{\mathbb{G}_T}, \Lambda_2^* \neq 1_{\mathbb{G}_T} \Rightarrow \lambda_1 \neq 0, \lambda_2 \neq 0$
- **Step 4.a:** If $\Big(\mathbf{e}(K_{3,i}, U_1) \cdot \mathbf{e}(K_{4,i}, U_2)\Big)^{v_1} = (\Lambda_1^*)^{v_i} \Rightarrow$

$$\mathbf{e}(g,g)^{v_1 \delta_1 \delta_2 (r_i' - r_i)} \cdot \mathbf{e}(h,g)^{\lambda_1 v_1 v_i} = \mathbf{e}(g,h)^{\lambda_1 v_1 v_i} \Rightarrow \mathbf{e}(g,g)^{v_1 \delta_1 \delta_2 (r_i' - r_i)} = 1_{\mathbb{G}_T}$$
$$\Rightarrow \forall i \in [n] : \; r_i = r_i' \Rightarrow K_{3,i} = g^{-\delta_2 r_i} \cdot g^{\lambda_1 v_i w_{2,i}}, K_{4,i} = g^{\delta_1 r_i} \cdot g^{-\lambda_1 v_i w_{1,i}}$$

And similar computations show that the equality in **"Step 4.b"** holds for all $i \in [n]$. Then we conclude that there exists $\phi_i \in \mathbb{Z}_p$ such that: $K_{5,i} = g^{-\theta_2 \phi_i} \cdot g^{\lambda_2 v_i t_{2,i}}, K_{6,i} = g^{\theta_1 \phi_i} \cdot g^{-\lambda_2 v_i t_{1,i}}$

– **Step 5**

$$K_A = g' \prod_{i=1}^{n} K_{3,i}^{-f_{1,i}} K_{4,i}^{-f_{2,i}} K_{5,i}^{-h_{1,i}} K_{6,i}^{-h_{2,i}} \iff$$

$$\iff \mathbf{e}(K_A, g) = \mathbf{e}(g' \prod_{i=1}^{n} K_{3,i}^{-f_{1,i}} K_{4,i}^{-f_{2,i}} K_{5,i}^{-h_{1,i}} K_{6,i}^{-h_{2,i}}, g)$$

$$\iff \mathbf{e}(K_A, g) = \Lambda \cdot \prod_{i=1}^{n} \mathbf{e}(K_{3,i}, F_{1,i})^{-1}.\mathbf{e}(K_{4,i}, F_{2,i})^{-1}.\mathbf{e}(K_{5,i}, H_{1,i})^{-1}\mathbf{e}(K_{6,i}, H_{2,i})^{-1}.$$

– **Step 6**

$$\prod_{i=1}^{n} \mathbf{e}(K_{3,i}, W_{1,i}) \cdot \mathbf{e}(K_{4,i}, W_{2,i}) \cdot \mathbf{e}(K_{5,i}, T_{1,i}) \cdot \mathbf{e}(K_{6,i}, T_{2,i}) = \mathbf{e}(h, K_B)^{-1}$$

$$= \prod_{i=1}^{n} \mathbf{e}(g^{-\delta_2 r_i}, g^{w_{1,i}}) \cdot \mathbf{e}(g^{\delta_1 r_i}, g^{w_{2,i}}) \cdot \mathbf{e}(g^{-\theta_2 \phi_i}, g^{t_{1,i}}) \cdot \mathbf{e}(g^{\theta_1 \phi_i}, g^{t_{2,i}})$$

$$= \prod_{i=1}^{n} \mathbf{e}(g^{r_i(\delta_1 w_{2,i} - \delta_2 w_{1,i})}, g) \cdot \mathbf{e}(g^{\phi_i(\theta_1 t_{2,i} - \theta_2 t_{1,i})}, g)$$

$$= \prod_{i=1}^{n} \mathbf{e}(g, h)^{r_i + \phi_i} = \mathbf{e}(h, K_B)^{-1} \Rightarrow K_B = \prod_{i=1}^{n} g^{-(r_i + \phi_i)}$$

The second relation is a disjunction of two predicates, $\mathcal{R}^{\mathsf{tok}}(x, w) = P_1^{\mathsf{tok}} \vee P_2^{\mathsf{tok}}$. The proof of membership for this relation can be implemented using the token verification algorithm for relation $\mathcal{R}_{\mathsf{IP}}^{\mathsf{tok}}$ 3 and assuming to have pairing product equations corresponding to the commitments in the two aforementioned predicates. We skip further details.

5.5 $\mathsf{NIWI}^{\mathsf{enc}} = \langle \mathcal{P}^{\mathsf{enc}}, \mathcal{V}^{\mathsf{enc}} \rangle$: NIWI-proof for encryption algorithm

For the relation $\mathcal{R}_{\mathsf{IP}}^{\mathsf{ct}}$, we first provide a proof of satisfiability for a system of equation related to a single ciphertext, that is $k = 1$, and we will later extend it to the case of two ciphertexts, that is $k = 2$.For $k > 2$, the algorithm is similar to the case $k = 2$.

Let $x = (\mathsf{mpk}, \mathsf{ct})$. We define the following variables, $(1 \le i \le n)$:

$$\begin{array}{llllll} \mathcal{S}_1 = g^{s_1} & ,\mathcal{S}_3 = g^{s_3} & ,\mathcal{S}_4 = g^{s_4} & ,\mathcal{X}_i = g^{x_i} & ,\mathcal{S}'_1 = g^{s'_1} & ,\mathcal{S}'_3 = g^{s'_3} \\ \mathcal{U}_1 = U_1^{s_3} & ,\mathcal{U}_2 = U_2^{s_3} & ,\mathcal{V}_1 = V_1^{s_4} & ,\mathcal{V}_2 = V_2^{s_4} & & \\ \mathcal{U}_1 = U_1^{s'_3} & ,\mathcal{U}_2 = U_2^{s'_3} & ,\mathcal{K}_1 = K_1^{s_2} & ,\mathcal{K}'_1 = K_1^{s'_2} & & \end{array}$$

We have the following Equations related to component ct_2:

$$\mathbf{e}(\mathsf{ct}_2, g) = \mathbf{e}(h^{s_1}, g) = \mathbf{e}(h, g^{s_1}) = \mathbf{e}(h, \mathcal{S}_1)$$

$$\mathbf{e}(\mathsf{ct}'_2, g) = \mathbf{e}(h^{s'_1}, g) = \mathbf{e}(h, g^{s'_1}) = \mathbf{e}(h, \mathcal{S}'_1),$$

and equations related $\mathsf{ct}_{j,i}$ for $j = 1, 2, 3, 4$ and $i = 1, \dots, n$:

$$\begin{aligned}
\mathbf{e}(\mathsf{ct}_{3,i}, g) = \mathbf{e}(W_{1,i}^{s_1}, g) \cdot \mathbf{e}(F_{1,i}^{s_2}, g) \cdot \mathbf{e}(U_1^{s_3 x_i}, g) &= \mathbf{e}(W_{1,i}, g^{s_1}) \cdot \mathbf{e}(F_{1,i}, g^{s_2}) \cdot \mathbf{e}(U_1^{s_3}, g^{x_i}) \\
&= \mathbf{e}(W_{1,i}, \mathcal{S}_1) \cdot \mathbf{e}(F_{1,i}, \mathsf{ct}_1) \cdot \mathbf{e}(\mathcal{U}_1, \mathcal{X}_i) \\
\Rightarrow \mathbf{e}(\mathsf{ct}_{3,i}, g) \cdot \mathbf{e}(F_{1,i}, \mathsf{ct}_1)^{-1} &= \mathbf{e}(W_{1,i}, \mathcal{S}_1) \cdot \mathbf{e}(\mathcal{U}_1, \mathcal{X}_i) \\
\mathbf{e}(\mathsf{ct}'_{3,i}, g) \cdot \mathbf{e}(F_{1,i}, \mathsf{ct}'_1)^{-1} &= \mathbf{e}(W_{1,i}, \mathcal{S}'_1) \cdot \mathbf{e}(\mathcal{U}'_1, \mathcal{X}_i) \\
\mathbf{e}(\mathsf{ct}_{5,i}, g) \cdot \mathbf{e}(H_{1,i}, \mathsf{ct}_2)^{-1} &= \mathbf{e}(T_{1,i}, \mathcal{S}_1) \cdot \mathbf{e}(\mathcal{V}_1, \mathcal{X}_i) \\
\mathbf{e}(\mathsf{ct}'_{5,i}, g) \cdot \mathbf{e}(H_{1,i}, \mathsf{ct}'_2)^{-1} &= \mathbf{e}(T_{1,i}, \mathcal{S}'_1) \cdot \mathbf{e}(\mathcal{V}'_1, \mathcal{X}_i) \\
\mathbf{e}(\mathsf{ct}_{6,i}, g) \cdot \mathbf{e}(H_{2,i}, \mathsf{ct}_2)^{-1} &= \mathbf{e}(T_{2,i}, \mathcal{S}_1) \cdot \mathbf{e}(\mathcal{V}_2, \mathcal{X}_i) \\
\mathbf{e}(\mathsf{ct}'_{6,i}, g) \cdot \mathbf{e}(H_{2,i}, \mathsf{ct}'_2)^{-1} &= \mathbf{e}(T_{2,i}, \mathcal{S}'_1) \cdot \mathbf{e}(\mathcal{V}_2, \mathcal{X}_i)
\end{aligned}$$

The equations show that the exponent of $U_b^{s_3}$ and $V_b^{s_4}$ in $\mathsf{ct}_{3,i}, \mathsf{ct}_{4,i}, \mathsf{ct}_{5,i}, \mathsf{ct}_{6,i}$ are x_i. So we have the following equation:

$$\begin{aligned}
\mathbf{e}(\mathcal{U}_1, U_2) \cdot \mathbf{e}(U_1^{-1}, \mathcal{U}_2) = \mathbf{e}(U^{s_3}, U_2) \cdot \mathbf{e}(U_1^{-1}, U_2^{s_3}) = \mathbf{e}(U_1, U_2)^{s_3 - s_3} = 1_{\mathbb{G}_T} \\
\mathbf{e}(\mathcal{V}_1, V_2) \cdot \mathbf{e}(V_1^{-1}, \mathcal{V}_2) = \mathbf{e}(V^{s_4}, V_2) \cdot \mathbf{e}(V_1^{-1}, V_2^{s_4}) = \mathbf{e}(V_1, V_2)^{s_4 - s_4} = 1_{\mathbb{G}_T}
\end{aligned}$$

The equation related to $\mathsf{ct}_7 = \mathbf{e}(g^{s_3}, g^{s_4})$ is the following:

$$\mathsf{ct}_7 = \mathbf{e}(g^{s_3}, g^{s_4}) = \mathbf{e}(\mathcal{S}_3, \mathcal{S}_4), \mathsf{ct}'_7 = \mathbf{e}(g^{s'_3}, g^{s_4}) = \mathbf{e}(\mathcal{S}'_3, \mathcal{S}_4)$$

To prove $s_3 \neq s'_3$, we just need check wether $\mathsf{ct}_7 \neq \mathsf{ct}'_7$ or not.

$$\mathsf{ct}_7 \neq \mathsf{ct}'_7 \Rightarrow \mathbf{e}(g^{s_3}, g^{s_4}) \neq \mathbf{e}(g^{s'_3}, g^{s_4}) \Rightarrow s_3 \neq s'_3.$$

The equation related to $\mathsf{ct}_7, \mathsf{ct}'_7$ is the following:

$$\begin{aligned}
\mathsf{ct}_8 = \Lambda^{-s_2} \cdot m, \mathsf{ct}'_8 = \Lambda^{-s'_2} \cdot m \Rightarrow \mathsf{ct}_8^{-1} \cdot \mathsf{ct}'_8 = \Lambda^{s_2} \cdot m^{-1} \Lambda^{-s'_2} \cdot m = \Lambda^{s_2 - s'_2} \\
\Rightarrow \mathsf{ct}_8^{-1} \cdot \mathsf{ct}'_8 = \mathbf{e}(K_1, K_2)^{s_2 - s'_2} = \mathbf{e}(K_1, K_2^{s_2}) \cdot \mathbf{e}(K_1^{-1}, K_2^{s'_2}) = \mathbf{e}(K_1, \mathcal{K}_2) \cdot \mathbf{e}(K_1^{-1}, \mathcal{K}'_1)
\end{aligned}$$

And to prove that the the power of Λ and g are both $-s_2$ and s_2 in ct_1 and ct_8, we add the following equation:

$$\mathbf{e}(\mathsf{ct}_1, K_1) = \mathbf{e}(g, \mathcal{K}_1), \mathbf{e}(\mathsf{ct}'_1, K_1) = \mathbf{e}(g, \mathcal{K}'_1)$$

So we have the following system of equation for one single ciphertext.

$$\mathsf{E_{ct}}: \begin{cases} \mathbf{e}(\mathsf{ct}_2, g) = \mathbf{e}(h, \mathcal{S}_1), \mathbf{e}(\mathsf{ct}'_2, g) = \mathbf{e}(h, \mathcal{S}'_1), \mathbf{e}(\hat{\mathsf{ct}}_2, \hat{g}) = \mathbf{e}(\hat{h}, \hat{\mathcal{S}}_1), \mathbf{e}(\hat{\mathsf{ct}}'_2, \hat{g}) = \mathbf{e}(\hat{h}, \hat{\mathcal{S}}'_1) \\ \mathbf{e}(\mathsf{ct}_{3,i}, g) \cdot \mathbf{e}(F_{1,i}, \mathsf{ct}_1)^{-1} = \mathbf{e}(W_{1,i}, \mathcal{S}_1) \cdot \mathbf{e}(\mathcal{U}_1, \mathcal{X}_i) \\ \mathbf{e}(\mathsf{ct}'_{3,i}, g) \cdot \mathbf{e}(F_{1,i}, \mathsf{ct}'_1)^{-1} = \mathbf{e}(W_{1,i}, \mathcal{S}'_1) \cdot \mathbf{e}(\mathcal{U}'_1, \mathcal{X}_i) \\ \mathbf{e}(\mathsf{ct}_{4,i}, g) \cdot \mathbf{e}(F_{2,i}, \mathsf{ct}_1)^{-1} = \mathbf{e}(W_{2,i}, \mathcal{S}_1) \cdot \mathbf{e}(\mathcal{U}_2, \mathcal{X}_i) \\ \mathbf{e}(\mathsf{ct}'_{4,i}, g) \cdot \mathbf{e}(F_{2,i}, \mathsf{ct}'_1)^{-1} = \mathbf{e}(W_{2,i}, \mathcal{S}'_1) \cdot \mathbf{e}(\mathcal{U}'_2, \mathcal{X}_i) \\ \mathbf{e}(\mathsf{ct}_{5,i}, g) \cdot \mathbf{e}(H_{1,i}, \mathsf{ct}_2)^{-1} = \mathbf{e}(T_{1,i}, \mathcal{S}_1) \cdot \mathbf{e}(\mathcal{V}_1, \mathcal{X}_i) \\ \mathbf{e}(\mathsf{ct}'_{5,i}, g) \cdot \mathbf{e}(H_{1,i}, \mathsf{ct}'_2)^{-1} = \mathbf{e}(T_{1,i}, \mathcal{S}'_1) \cdot \mathbf{e}(\mathcal{V}'_1, \mathcal{X}_i) \\ \mathbf{e}(\mathsf{ct}_{6,i}, g) \cdot \mathbf{e}(H_{2,i}, \mathsf{ct}_2)^{-1} = \mathbf{e}(T_{2,i}, \mathcal{S}_1) \cdot \mathbf{e}(\mathcal{V}_2, \mathcal{X}_i) \\ \mathbf{e}(\mathsf{ct}'_{6,i}, g) \cdot \mathbf{e}(H_{2,i}, \mathsf{ct}'_2)^{-1} = \mathbf{e}(T_{2,i}, \mathcal{S}'_1) \cdot \mathbf{e}(\mathcal{V}'_2, \mathcal{X}_i) \\ \mathsf{ct}_7 = \mathbf{e}(\mathcal{S}_3, \mathcal{S}_4), \mathsf{ct}'_7 = \mathbf{e}(\mathcal{S}'_3, \mathcal{S}_4), \hat{\mathsf{ct}}_7 = \mathbf{e}(\hat{\mathcal{S}}_3, \hat{\mathcal{S}}_4), \hat{\mathsf{ct}}'_7 = \mathbf{e}(\hat{\mathcal{S}}'_3, \hat{\mathcal{S}}_4) \\ \mathsf{ct}_8^{-1} \cdot \mathsf{ct}'_8 = \mathbf{e}(K_1, \mathcal{K}_2) \cdot \mathbf{e}(K_1^{-1}, \mathcal{K}'_1), \hat{\mathsf{ct}}_8^{-1} \cdot \hat{\mathsf{ct}}'_8 = \mathbf{e}(\hat{K}_1, \hat{\mathcal{K}}_2) \cdot \mathbf{e}(\hat{K}_1^{-1}, \hat{\mathcal{K}}'_1) \\ \mathbf{e}(\mathsf{ct}_1, K_1) = \mathbf{e}(g, \mathcal{K}_1), \mathbf{e}(\mathsf{ct}'_1, K_1) = \mathbf{e}(g, \mathcal{K}'_1) \end{cases}$$

Now we need to provide a proof that two ciphertext $\mathsf{ct}, \hat{\mathsf{ct}}$ are the encryption of a single message m and a single attribute $\boldsymbol{x}$:

$$\mathcal{X}_i = g^{x_i}, \hat{\mathcal{X}}_i = \hat{g}^{x_i} \Rightarrow \mathbf{e}(\mathcal{X}_i, \hat{g}) = \mathbf{e}(g, \hat{\mathcal{X}}_i) \Rightarrow \mathbf{e}(\mathcal{X}_i, \hat{g}) \cdot \mathbf{e}(g, \hat{\mathcal{X}}_i)^{-1} = 1_{\mathbb{G}_T}$$

Notice that $\mathsf{ct}_8, \mathsf{ct}'_8$ are the only components of te ciphertext which are related to the message and $\mathsf{ct}_1, \mathsf{ct}'_1$ are the only components which are related to them, so we have:

$$\mathsf{ct}_8 = \Lambda^{-s_2} \cdot m, \qquad \mathsf{ct}'_8 = \Lambda^{-s'_2} \cdot m$$
$$\hat{\mathsf{ct}}_8 = \hat{\Lambda}^{-\hat{s}_2} \cdot m, \qquad \hat{\mathsf{ct}}_8{}' = \hat{\Lambda}^{-\hat{s}'_2} \cdot m.$$

$$\left.\begin{matrix} \mathsf{ct}_8 = \Lambda^{-s_2}\mathbf{m} \\ \hat{\mathsf{ct}}_8 = \hat{\Lambda}^{-\hat{s}_2}\mathbf{m} \end{matrix}\right\} \Rightarrow \mathsf{ct}_7\hat{\mathsf{ct}}_7^{-1} = \Lambda^{-s_2} \cdot \hat{\Lambda}^{\hat{s}_2} =$$

$$= \mathbf{e}(K_1^{s_2}, K_2^{-1}) \cdot \mathbf{e}(\hat{K}_1^{\hat{s}_2}, \hat{K}_2) = \mathbf{e}(K_1^{-1}, K_2^{s_2}) \cdot \mathbf{e}(\hat{K}_1, \hat{K}_2^{\hat{s}_2})$$
$$= \mathbf{e}(\mathcal{K}_1, K_2^{-1},) \cdot \mathbf{e}(\hat{\mathcal{K}}_1, \hat{K}_2) = \mathbf{e}(K_1^{-1}, \mathcal{K}_2) \cdot \mathbf{e}(\hat{K}_1, \hat{\mathcal{K}}_2)$$

So the prover has to provide a proof for the following system of equations:

$$\mathsf{E}_{\mathsf{ct}-\hat{\mathsf{ct}}}: \begin{cases} \mathsf{ct}_8\hat{\mathsf{ct}}_8^{-1} = \mathbf{e}(\mathcal{K}_1, K_2^{-1},) \cdot \mathbf{e}(\hat{\mathcal{K}}_1, \hat{K}_2) \\ \mathsf{ct}_8\hat{\mathsf{ct}}_8^{-1} = \mathbf{e}(K_1^{-1}, \mathcal{K}_2) \cdot \mathbf{e}(\hat{K}_1, \hat{\mathcal{K}}_2) \\ \mathbf{e}(g, \mathcal{K}_1) = \mathbf{e}(\mathsf{ct}_1, K_1) \\ \mathbf{e}(\hat{g}, \hat{\mathcal{K}}_1) = \mathbf{e}(\hat{\mathsf{ct}}_1, \hat{K}_1) \\ \mathbf{e}(\mathcal{X}_i, \hat{g}) \cdot \mathbf{e}(g, \hat{\mathcal{X}}_i)^{-1} = 1_{\mathbb{G}_T} \end{cases}$$

Summing up, the NIWI-proof system for encryption algorithm, would be same as GS NIWI-proof system which takes as input pairing product equations $\mathsf{E_{ct}}$ and $\mathsf{E}_{\mathsf{ct}-\hat{\mathsf{ct}}}$.

6 Conclusion

Our main contribution in this paper is the construction of the first *efficient* verifiable (attribute-hiding) inner product encryption scheme from bilinear groups. The privacy of our scheme is based on the standard DLIN assumption whereas its verifiability is unconditional. Towards this goal, we also construct the first perfectly correct (attribute-hiding) inner product encryption scheme for plaintexts of arbitrary length. Our verifiable inner product encryption scheme is selectively secure only; we leave as an interesting open problem the construction a fully secure one.

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. Cryptology ePrint Archive, Report 2005/254 (2005), `http://eprint.iacr.org/2005/254`
2. Badrinarayanan, S., Goyal, V., Jain, A., Sahai, A.: Verifiable functional encryption. In: Proceedings, Part II, of the 22Nd International Conference on Advances in Cryptology — ASIACRYPT 2016 - Volume 10032. pp. 557–587. Springer-Verlag New York, Inc., New York, NY, USA (2016)
3. 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.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10401, pp. 67–98. Springer (2017). https://doi.org/10.1007/978-3-319-63688-7_3, `https://doi.org/10.1007/978-3-319-63688-7_3`
4. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J. (eds.) Advances in Cryptology – EUROCRYPT 2004. Lecture Notes in Computer Science, vol. 3027, pp. 506–522. Springer (May 2004)
5. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) Advances in Cryptology – CRYPTO 2001. Lecture Notes in Computer Science, vol. 2139, pp. 213–229. Springer (Aug 2001)
6. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Ishai, Y. (ed.) TCC 2011: 8th Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 6597, pp. 253–273. Springer (Mar 2011)
7. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007: 4th Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 4392, pp. 535–554. Springer (Feb 2007)
8. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) 8th IMA International Conference on Cryptography and Coding. Lecture Notes in Computer Science, vol. 2260, pp. 360–363. Springer (Dec 2001)
9. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) Advances in Cryptology – ASIACRYPT 2006. Lecture Notes in Computer Science, vol. 4284, pp. 444–459. Springer (Dec 2006)

10. Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive zaps and new techniques for NIZK. In: Dwork, C. (ed.) Advances in Cryptology – CRYPTO 2006. Lecture Notes in Computer Science, vol. 4117, pp. 97–111. Springer (Aug 2006)
11. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) Advances in Cryptology – EUROCRYPT 2008. Lecture Notes in Computer Science, vol. 4965, pp. 415–432. Springer (Apr 2008)
12. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007. pp. 21–30 (2007)
13. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) Advances in Cryptology – ASIACRYPT 2010. Lecture Notes in Computer Science, vol. 6477, pp. 177–194. Springer (Dec 2010)
14. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N.P. (ed.) Advances in Cryptology – EUROCRYPT 2008. Lecture Notes in Computer Science, vol. 4965, pp. 146–162. Springer (Apr 2008)
15. Lewko, A.B., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. Lecture Notes in Computer Science, vol. 6110, pp. 62–91. Springer (May 2010)
16. Libert, B., Ramanna, S.C., Yung, M.: Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In: 43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy. pp. 30:1–30:14 (2016)
17. Okamoto, T., Takashima, K.: Adaptively attribute-hiding (hierarchical) inner product encryption. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012. Lecture Notes in Computer Science, vol. 7237, pp. 591–608. Springer (Apr 2012)
18. Park, J.H.: Inner-product encryption under standard assumptions. Des. Codes Cryptography **58**(3), 235–257 (2011). https://doi.org/10.1007/s10623-010-9405-9, `https://doi.org/10.1007/s10623-010-9405-9`
19. Sahai, A., Waters, B.R.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) Advances in Cryptology – EUROCRYPT 2005. Lecture Notes in Computer Science, vol. 3494, pp. 457–473. Springer (May 2005)
20. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) Advances in Cryptology – CRYPTO'84. Lecture Notes in Computer Science, vol. 196, pp. 47–53. Springer (Aug 1984)