CCA-secure Predicate Encryption from Pair Encoding in Prime Order Groups: Generic and Efficient

Sanjit Chatterjee, Sayantan Mukherjee, and Tapas Pandit

Indian Institute of Science, Bangalore
{sanjit,sayantan.mukherjee}@csa.iisc.ernet.in,tapasgmmath@gmail.com

Abstract. Attrapadung (Eurocrypt 2014) proposed a generic framework called pair encoding to simplify the design and proof of security of CPA-secure predicate encryption (PE) instantiated in composite order groups.Later Attrapadung (Asiacrypt 2016) extended this idea in prime order groups. Yamada et al. (PKC 2011, PKC 2012) and Nandi et al. (ePrint Archive: 2015/457, AAECC 2017) proposed generic conversion frameworks to achieve CCA-secure PE from CPA-secure PE provided the encryption schemes have properties like delegation or verifiability. The delegation property is harder to achieve and verifiability based conversion degrades the decryption performance due to a high number of additional pairing evaluations. Blömer et al. (CT-RSA 2016) proposed a direct fully CCA-secure predicate encryption in composite order groups but it was less efficient as it needed a large number of pairing evaluations to check ciphertext consistency. As an alternative, Nandi et al. (ePrint Archive: 2015/955) proposed a direct conversion technique in composite order groups. We extend the direct conversion technique of Nandi et al. in the prime order groups on the CPA-secure PE construction by Attrapadung (Asiacrypt 2016) and prove our scheme to be CCA-secure in a quite different manner. Our conversion technique incurs a cost of exactly three additional ciphertext components and only one additional unit pairing evaluation during decryption. This is a significant improvement over the available conversion mechanisms in prime order groups. We also have presented an alternative construction of direct CCA-secure predicate encryption scheme which is more efficient in the ciphertext size (only one additional ciphertext component) at the cost of increase in pairing evaluations (three additional unit precisely) required during decryption.

1 Introduction

The concept of Identity-Based Encryption (IBE) was proposed by Shamir [Sha85] as a replacement of traditional public key encryption. In an IBE, a sender can encrypt a message for a receiver by knowing only the identity of the receiver along with some system information. The notion of IBE thus gets rid of the burden of distributing correct public key of each user completely.

The first-ever construction of IBE was due to Boneh and Franklin [BF01]. Informally, in an IBE system, a ciphertext $\mathbf{C}$ created for a user of identity y can only be decrypted by that particular user (i.e. having identity y). One can view IBE as a realization of *equality predicate* in the encrypted domain. Subsequently IBE was generalized to emulate *access control predicate* (Attribute-Based Encryption [GPSW06,BSW07,Wat11]), *inner product predicate* (Inner Product Encryption [KSW08,OT09,SSW09]), *vector containment predicate* (Spatial Encryption [BH08]) or *subspace predicate* (Doubly Spatial Encryption [Ham11]) etc. All the aforementioned schemes are behaviorally similar in the sense they all emulate certain predicate function such that a receiver can decrypt a ciphertext if (s)he *satisfies* the predicate function. Such schemes can be viewed as different cases of the class *predicate encryption* (PE). A predicate encryption is defined by a predicate tuple $(\mathcal{X}, \mathcal{Y}, R)$ where $\mathcal{X}$ and $\mathcal{Y}$ are key-space and data-space respectively and $R : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ is the predicate function. A user having a secret key ($\mathbf{K}$) with respect to the key-index x can decrypt a ciphertext ($\mathbf{C}$) encrypted under a data-index y if $R(x, y) = 1$.

The dual system technique [Wat09,LW10] allows one to construct adaptively secure predicate encryption schemes. Attrapadung [Att14] and Wee [Wee14] independently observed a similarity in the structure of the proofs of dual system technique based adaptively secure predicate encryption schemes. The notions of *pair encoding* [Att14] and *predicate encoding* [Wee14] were introduced as abstraction of complex key and ciphertext structure of available predicate encryptions. Such encodings allowed them to construct adaptively CPA-secure predicate encryptions using dual system technique. This new approach not only allowed them to improve the performance of several available predicate encryption schemes but also to instantiate several completely new schemes. For example, pair encoding [Att14]

allowed the first-ever construction of PE for regular language, ABE with constant-size ciphertext etc. However, the CPA-secure predicate encryption instantiation was performed in composite order groups.

Later Attrapadung [Att16] and Chen et al. [CGW15] constructed adaptive CPA-secure predicate encryption schemes in the prime order groups using pair encoding and predicate encoding respectively. Even though both the constructions were in prime order groups, [CGW15] was even more modular due to the use of *dual system group* (DSG) [CW14]. Agrawal et al. [AC16,AC17] integrated pair encoding and dual system group and introduced different security notions for pair encoding. The authors in [AC16,ABS16] noted that pair encoding is more general than predicate encoding. However, all these schemes aimed at constructing CPA-secure predicate encryption.

Motivation. In various practical scenario, CCA-security is assumed to be mandatory. One can use available generic techniques [YAHK11,YAS+12,NP15a,NP17] to convert CPA-secure predicate encryption into CCA-secure predicate encryption. Informally these techniques add new components in the CPA-ciphertext that can be used later to check if the ciphertext has been tampered in the line. Therefore they face problems of two-fold – (1) increased length of key-indices and data-indices which result in a *bigger* secret key and ciphertext and (2) extra cost to perform verifiability or delegation. For example, verifiability based solution makes the decryption lot costlier than the cost of decryption in the CPA-secure scheme in terms of the number of pairings evaluated (See Appendix A). This degrades the performance as the decryption algorithm has to perform additional all most twice of $m_1 \times w_1 \times (d^2 + d) \times (m_2 + 1)$ many pairing evaluations. Blömer et al. [BL16] proposed a direct CCA-secure predicate encryption from pair encodings in composite order groups. The verifiability based check, that they used, takes nearly the cost of CPA-decryption number of pairing evaluations, thus degrades the performance of the decryption. Recently Nandi et al. [NP15b] suggested a direct conversion to CCA-secure predicate encryptions from pair encodings. Even though that conversion is efficient and generic, it is provided in the composite order group. Naturally one would like to construct a direct CCA-secure predicate encryption in prime order groups from a CPA-secure predicate encryption without compromising the performance.

Our Contribution. As pair encoding is more general than predicate encoding schemes, we use pair encoding scheme based generic construction of CPA-secure predicate encryption [Att16] to construct a direct adaptive CCA-secure predicate encryption in prime order groups that does not compromise the performance. The ciphertext in our construction adds *exactly three* additional components to the ciphertext of [Att16] namely a $(d+1)$-tuple made up of source group elements (i.e. an element of $\mathbb{G}_1^{(d+1)}$), an OTS verification key and a signature. During decryption our construction needs *only one* additional unit[1] of pairing evaluation along with an OTS verification.

As an alternative, we have presented another direct CCA-secure predicate encryption construction (in Appendix C) without using the primitive OTS. It adds *only one* additional component to the ciphertext of [Att16] at the cost of *three* additional unit pairing evaluations during decryption.

Organization of the Paper. Section 2 contains necessary definitions and notations that is followed in this paper. In Section 3, the definition of pair encoding scheme with its security is recalled. In Section 4 we describe our generic conversion mechanism to achieve cca-security. Section 5 proves the security of our construction. Section 6 concludes the paper. We describe standard CPA-to-CCA conversion in Appendix A to explain the cost of such conversion. The games in the security argument that we mimic from [Att16] is presented in Appendix B. We present a (some-what) less efficient direct CCA-secure predicate encryption construction in Appendix C.

2 Preliminaries

Notations. We denote $[a, b] = \{i \in \mathbb{N} : a \leq i \leq b\}$ and $[n] = [1, n]$. We assume $\boldsymbol{v}$ is a vector having components $v_1, \ldots, v_n$. By $s \xleftarrow{\$} S$ we denote a uniformly random choice s from S. 1^λ denotes the security parameter for $\lambda \in \mathbb{N}$. Any $\mathbf{x} \in X^t$ is a t-dimensional column vector whereas we use $\mathbf{x} \in X^{1\times t}$ and $\mathbf{x} \in \left(X\right)^t$ to denote t-dimensional row vectors.

[1] By one unit we mean $(d+1)$ many pairing evaluation for a $(d+1)$-dimensional system.

Predicate Family. The predicate family for an index family $\varkappa$ is $\mathcal{R} = \{R_\kappa\}_{\kappa \in \varkappa}$, where $R_\kappa : \mathcal{X}_\kappa \times \mathcal{Y}_\kappa \to \{0,1\}$ is a predicate function and $\mathcal{X}_\kappa$ and $\mathcal{Y}_\kappa$ are key-space and data-space respectively. We will often omit κ in the subscript for the simplicity of representation.

2.1 Predicate Encryption

A predicate encryption (PE) scheme Π_R for predicate function $R : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ consists of following algorithms.

- $\mathsf{Setup}(1^\lambda, \kappa)$ for the security parameter $\lambda \in \mathbb{N}$ generates master secret key msk and public key mpk.
- $\mathsf{KeyGen}(msk, x)$ generates secret key $\mathbf{K}$ of the given key-index $x \in \mathcal{X}$.
- $\mathsf{Enc}(mpk, y, M)$ takes as input data-index $y \in \mathcal{Y}$ and a message $M \in \mathcal{M}$ and generates ciphertext $\mathbf{C}$.
- $\mathsf{Dec}(\mathbf{K}, \mathbf{C})$ takes a key $\mathbf{K}$ corresponding to key-index x and a ciphertext $\mathbf{C}$ corresponding to data-index y and outputs a message M or $\perp$.

Correctness. A predicate encryption scheme is said to be correct if for all $(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda, \kappa)$, all $y \in \mathcal{Y}$, all $M \in \mathcal{M}$, all $\mathbf{C} \leftarrow \mathsf{Enc}(mpk, y, M)$, all $x \in \mathcal{X}$, all $\mathbf{K} \leftarrow \mathsf{KeyGen}(msk, x)$,

$$\mathsf{Dec}(\mathbf{K}, \mathbf{C}) = \begin{cases} M & \text{if } R(x,y) = 1 \\ \perp & \text{if } R(x,y) = 0 \end{cases}.$$

Security. Chosen ciphertext security (IND-CCA) of a predicate encryption scheme Π_R can be modeled as a security game between challenger $\mathcal{C}$ and adversary $\mathcal{A}$.

- Setup: $\mathcal{C}$ gives out mpk and keeps msk as secret.
- $\mathsf{Phase\text{-}I\ Query}$: Queries are performed to available oracles as follows.
  - $\mathsf{Key\ Query}$: Given a key-index x, keygen oracle $\mathcal{O}_K$ returns $\mathbf{K} \leftarrow \mathsf{KeyGen}(msk, x)$.
  - $\mathsf{Dec\ Query}$: Given $(x, \mathbf{C})$, decryption oracle $\mathcal{O}_D$ returns $\mathsf{Dec}(\mathbf{K}, \mathbf{C})$.
- $\mathsf{Challenge}$: $\mathcal{A}$ provides challenge data-index y^* (such that $R(x, y^*) = 0$ for all key query x) and two messages (M_0, M_1) of equal length. $\mathcal{C}$ generates $\mathbf{C}^* \leftarrow \mathsf{Enc}(mpk, y^*, M_\mathfrak{b})$ for $\mathfrak{b} \xleftarrow{\$} \{0,1\}$.
- $\mathsf{Phase\text{-}II\ Query}$: Queries are performed to available oracles as follows.
  - $\mathsf{Key\ Query}$: Given a key-index x such that $R(x, y^*) = 0$, keygen oracle $\mathcal{O}_K$ returns $\mathbf{K} \leftarrow \mathsf{KeyGen}(msk, x)$.
  - $\mathsf{Dec\ Query}$: Given $(x, \mathbf{C})$, decryption oracle $\mathcal{O}_D$ returns $\mathsf{Dec}(\mathbf{K}, \mathbf{C})$ if $R(x, y^*) = 1$ and $\mathbf{C} = \mathbf{C}^*$ does not happen at the same time.
- Guess: $\mathcal{A}$ outputs its guess $\mathfrak{b}' \in \{0,1\}$ and wins if $\mathfrak{b} = \mathfrak{b}'$.

For any adversary $\mathcal{A}$ the advantage is defined as follows.

$$\mathbf{Adv}_{\mathcal{A}}^{\Pi_R}(\lambda) = |\Pr[\mathfrak{b} = \mathfrak{b}'] - 1/2|.$$

A predicate encryption scheme is said to be IND-CCA secure if for any efficient adversary $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{\Pi_R}(\lambda) \leq \mathrm{neg}(\lambda)$. In this paper we use the advantage notation $\mathbf{Adv}_{\mathcal{A}}^{\Pi_R}(\lambda)$ to denote the advantage of any adversary $\mathcal{A}$ to break the predicate encryption scheme Π_R. If the decryption oracle is not available to the adversary, we call such security model as IND-CPA security model.

2.2 Matrix Diffie-Hellman Problem

We call $\mathcal{D}_d$ a matrix distribution if it outputs matrices in $\mathbb{Z}_p^{(d+1)\times(d+1)}$ of the form $\mathbf{T} = \begin{pmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{c} & 1 \end{pmatrix}$ such that $\mathbf{M} \in \mathbb{GL}_{p,d}$. We say that $\mathcal{D}_d$-*Matrix Diffie-Hellman Assumption* holds in $\mathcal{G}$ if for any PPT adversary $\mathcal{A}$, the advantage,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathcal{D}_d\text{-MatDH}}(\lambda) = \left| \Pr\left[\mathcal{A}(\mathbf{G}, g_1^{\mathbf{T}}, g_1^{\mathbf{T}\begin{pmatrix} \boldsymbol{y} \\ 0 \end{pmatrix}}) = 1\right] - \Pr\left[\mathcal{A}(\mathbf{G}, g_1^{\mathbf{T}}, g_1^{\mathbf{T}\begin{pmatrix} \boldsymbol{y} \\ \hat{y} \end{pmatrix}}) = 1\right] \right| \leq \mathrm{neg}(\lambda)$$

where the probability is taken over $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p) \xleftarrow{\$} \mathcal{G}(\lambda)$, $(g_1, g_2) \xleftarrow{\$} \mathbb{G}_1 \times \mathbb{G}_2$, $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$, $\boldsymbol{y} \xleftarrow{\$} \mathbb{Z}_p^d$, $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$ and the internal randomness of $\mathcal{A}$ such that $\mathbf{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, g_1, g_2)$. It is to be noted that *Matrix Diffie-Hellman*

Problem is random self reducible [Att16,EHK+17]. Therefore given an instance of the problem, one can construct polynomial number of different instances of that problem without degrading the reduction i.e. given $\left(g_1^{\mathbf{T}}, g_1^{\mathbf{T}\begin{pmatrix}\mathbf{y}\\ \hat{\boldsymbol{y}}\end{pmatrix}}\right)$ it is easy to construct $\left(g_1^{\mathbf{T}}, g_1^{\mathbf{T}\begin{pmatrix}\mathbf{Y}\\ \hat{\boldsymbol{y}}\end{pmatrix}}\right)$ such that $\mathbf{Y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m)$ is uniformly random in $\left(\mathbb{Z}_p^d\right)^m$, $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_m)$ is uniformly random in $\left(\mathbb{Z}_p\right)^m$ for $m = \mathsf{poly}(\lambda)$.

2.3 Parameter Hiding Lemma

The parameter-hiding lemma defined by [Att16, Lemma 2] is the following. Given g_1, g_2, $\mathbf{B}$, $\mathbf{Z}$, $g_1^{\mathbf{H}_i\mathbf{B}\begin{pmatrix}\mathbf{I}_d\\ 0\end{pmatrix}} \in \mathbb{G}_1^{(d+1)\times d}$ and $g_2^{\mathbf{H}_i^\top\mathbf{Z}\begin{pmatrix}\mathbf{I}_d\\ 0\end{pmatrix}} \in \mathbb{G}_2^{(d+1)\times d}$, the $(d+1, d+1)^{th}$ entry of the matrix $\mathbf{B}^{-1}\mathbf{H}_i\mathbf{B}$ is information-theoretically hidden.

2.4 Strong One-Time Signature

Signature Scheme. A signature scheme consists of three PPTs,

- Gen outputs verification key vk and signing key sk.
- Sign computes a signature σ on the input message m.
- Verify on (m', σ') input it outputs 1 if σ' is a valid signature on m'.

Security Definition. Strong One-Time Signature (OTS) model is defined by the game between challenger $\mathcal{C}$ and adversary $\mathcal{A}$ as follows.

- **Gen**: $\mathcal{C}$ runs $(vk, sk) \leftarrow \mathsf{OTS.Gen}(1^\lambda)$. $\mathcal{A}$ is provided with vk.
- **Query**: $\mathcal{A}$ is given access to oracle $\mathsf{OTS.Sign}(sk, \cdot)$ for only one query. Let $\mathcal{A}$ queries with a message m and gets back a signature σ.
- **Forge**: $\mathcal{A}$ outputs a pair (m^*, σ^*).

$\mathcal{A}$ wins this game if $\mathsf{OTS.Verify}(vk, m^*, \sigma^*) = 1$ and $(m, \sigma) \neq (m^*, \sigma^*)$. The advantage of adversary $\mathcal{A}$ is defined to be the probability of its win and is denoted by $\mathsf{Adv}^{\text{sUf-CMA}}_{\mathcal{A},\mathsf{OTS}}(\lambda)$. We call a signature one-time secure if for any efficient adversary $\mathcal{A}$, $\mathsf{Adv}^{\text{sUf-CMA}}_{\mathcal{A},\mathsf{OTS}}(\lambda) \leq \mathrm{neg}(\lambda)$.

3 Pair Encoding Schemes

Attrapadung [Att14] introduced the notion of pair encoding scheme. Later he [Att16] refined the notion with *regularity* of encoding. Here we recall the definition of pair encoding. We will describe the *regularity properties* of pair encoding in Section 4.1 (precisely Properties $\mathcal{P}1, \mathcal{P}2, \mathcal{P}3, \mathcal{P}4$). It should be noted that all the available predicate encryption schemes constructed on some encoding schemes [Att14,Wee14,CGW15,AC16,Att16,AC17] follow the regularity of encoding property.

A Pair Encoding P for a predicate function $R_\kappa : \mathcal{X}_\kappa \times \mathcal{Y}_\kappa \to \{0, 1\}$ indexed by $\kappa = (N \in \mathbb{N}, \mathrm{par})$ consists of four deterministic algorithms:

- $\mathsf{Param}(\kappa) \to n$ where n is the number of *common variables* $\mathbf{h} = (h_1, \ldots, h_n)$ in EncK and EncC.

– $\mathsf{EncK}(x, N) \to (\mathbf{k} = (k_1, \ldots, k_{m_1}); m_2)$ where each k_ι for $\iota \in [m_1]$ is a polynomial of m_2 own variables $\mathbf{r} = (r_1, \ldots, r_{m_2})$, common variables $\mathbf{h}$ and α. Each polynomial k_ι is

$$k_\iota(\alpha, \mathbf{r}, \mathbf{h}) = b_\iota \alpha + \sum_{j \in [m_2]} b_{\iota j} r_j + \sum_{\substack{j \in [m_2] \\ k \in [n]}} b_{\iota jk} r_j h_k,$$

where $b_\iota, b_{\iota j}, b_{\iota jk} \in \mathbb{Z}_N$ for all $\iota \in [m_1]$, all $j \in [m_2]$ and all $k \in [n]$.

– $\mathsf{EncC}(y, N) \to (\mathbf{c}_y = (c_1, \ldots, c_{w_1}); w_2)$ where each $c_{\tilde{\iota}}$ for $\tilde{\iota} \in [w_1]$ is a polynomial of $(w_2 + 1)$ own variables $\mathbf{s} = (s_0, \ldots, s_{w_2})$ and common variables $\mathbf{h}$. Each polynomial $c_{\tilde{\iota}}$ is

$$c_{\tilde{\iota}}(\mathbf{s}, \mathbf{h}) = \sum_{j \in [0, w_2]} a_{\tilde{\iota} j} s_j + \sum_{\substack{j \in [0, w_2] \\ k \in [n]}} a_{\tilde{\iota} jk} s_j h_k,$$

where $a_{\tilde{\iota} j}, a_{\tilde{\iota} jk} \in \mathbb{Z}_N$ for all $\tilde{\iota} \in [w_1]$, all $j \in [0, w_2]$ and all $k \in [n]$.

– $\mathsf{Pair}(x, y, N) \to \mathbf{E} \in \mathbb{Z}_N^{m_1 \times w_1}$.

Correctness. A pair encoding scheme is said to be correct if for all $N \in \mathbb{N}$, for all $y \in \mathcal{Y}_\kappa$, $\mathbf{c} \leftarrow \mathsf{EncC}(y, N)$, all $x \in \mathcal{X}_\kappa$, $\mathbf{k} \leftarrow \mathsf{EncK}(x, N)$ and $\mathbf{E} \leftarrow \mathsf{Pair}(x, y, N)$, $\mathbf{k}\mathbf{E}\mathbf{c}^\top = \alpha s_0$ if $R(x, y) = 1$.

Properties of Pair Encoding Schemes. We define two natural properties of the pair encoding scheme as follows.

– Param-Vanishing: $\mathbf{k}(\alpha, \mathbf{0}, \mathbf{h}) = \mathbf{k}(\alpha, \mathbf{0}, \mathbf{0})$.
– Linearity:

$$\mathbf{k}(\alpha_1, \mathbf{r}_1, \mathbf{h}) + \mathbf{k}(\alpha_2, \mathbf{r}_2, \mathbf{h}) = \mathbf{k}(\alpha_1 + \alpha_2, \mathbf{r}_1 + \mathbf{r}_2, \mathbf{h})$$
and
$$\mathbf{c}(\mathbf{s}_1, \mathbf{h}) + \mathbf{c}(\mathbf{s}_2, \mathbf{h}) = \mathbf{c}(\mathbf{s}_1 + \mathbf{s}_2, \mathbf{h}).$$

3.1 Security Definitions for Pair Encoding Schemes

Perfect Security. Pair Encoding P is said to be *perfectly master-key hiding* if the following holds. Suppose $R(x, y) = 0$. Let $n \leftarrow \mathsf{Param}(\kappa)$, $\mathbf{k} \leftarrow \mathsf{EncK}(x, N)$ and $\mathbf{c} \leftarrow \mathsf{EncC}(y, N)$. Then the following distributions,

$$\{\mathbf{c}(\mathbf{s}, \mathbf{h}), \mathbf{k}(0, \mathbf{r}, \mathbf{h})\} \text{ and } \{\mathbf{c}(\mathbf{s}, \mathbf{h}), \mathbf{k}(\alpha, \mathbf{r}, \mathbf{h})\}$$

are identical where $\mathbf{h} \xleftarrow{\$} \mathbb{Z}_N^n$, $\alpha \xleftarrow{\$} \mathbb{Z}_N$, $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_N^{m_2}$ and $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_N^{(w_2+1)}$.

Computational Security. Two types of computational security notions CMH and SMH are defined in [Att16] for a bilinear group generator $\mathcal{G}$. We use these security notions to argue indistinguishability of *type-1* and *type-2* semi-functional keys. For the sake of completeness we note these notions of security down here.

Both the security notions (CMH and SMH) are defined as the security games as follows.

$$\begin{aligned}\mathrm{Exp}_{\mathcal{G},\mathsf{G},\mathfrak{d},\alpha,t_1,t_2}(\lambda) : (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, N) \leftarrow \mathcal{G}(\lambda);\ (g_1, g_2) \xleftarrow{\$} \mathbb{G}_1 \times \mathbb{G}_2,\\ \alpha \xleftarrow{\$} \mathbb{Z}_N, n \leftarrow \mathsf{Param}(\kappa), \mathbf{h} \xleftarrow{\$} \mathbb{Z}_N^n;\\ \mathrm{st} \leftarrow \mathcal{A}_1^{\mathcal{O}^1_{\mathsf{G},\mathfrak{d},\alpha,\mathbf{h}}(\cdot)}(g_1, g_2);\ \mathfrak{d}' \leftarrow \mathcal{A}_2^{\mathcal{O}^2_{\mathsf{G},\mathfrak{d},\alpha,\mathbf{h}}(\cdot)}(\mathrm{st})\end{aligned}$$

where $\mathsf{G} \in \{\mathsf{CMH}, \mathsf{SMH}\}$ and each oracle $\mathcal{O}^1, \mathcal{O}^2$ can be queried at most t_1, t_2 times respectively.

– CMH:
  • $\mathcal{O}^1_{\mathsf{CMH},\mathfrak{d},\alpha,\mathbf{h}}(x^*)$: Run $(\mathbf{k}; m_2) \leftarrow \mathsf{EncK}(x^*)$; $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_N^{m_2}$;
  $$\text{return } \mathbf{V} \leftarrow \begin{cases} g_2^{\mathbf{k}(0,\mathbf{r},\mathbf{h})} & \text{if } \mathfrak{d} = 0 \\ g_2^{\mathbf{k}(\alpha,\mathbf{r},\mathbf{h})} & \text{if } \mathfrak{d} = 1 \end{cases}.$$
  • $\mathcal{O}^2_{\mathsf{CMH},\mathfrak{d},\alpha,\mathbf{h}}(y)$: If $R(x^*, y) = 1$, then return $\perp$.
  $$\text{Else run } (\mathbf{c}; w_2) \leftarrow \mathsf{EncC}(y);\ \mathbf{s} \xleftarrow{\$} \mathbb{Z}_N^{(w_2+1)};$$
  $$\text{return } \mathbf{U} \leftarrow g_1^{\mathbf{c}(\mathbf{s},\mathbf{h})}.$$

– SMH:
  - $\mathcal{O}^1_{\mathsf{SMH},\mathfrak{d},\alpha,\mathbf{h}}(y^*)$: Run $(\mathbf{c}; w_2) \leftarrow \mathsf{EncC}(y^*)$; $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_N^{(w_2+1)}$;
    return $\mathbf{U} \leftarrow g_1^{\mathbf{c}(\mathbf{s},\mathbf{h})}$.
  - $\mathcal{O}^2_{\mathsf{SMH},\mathfrak{d},\alpha,\mathbf{h}}(x)$: If $R(x, y^*) = 1$, then return $\perp$.
    Else run $(\mathbf{k}; m_2) \leftarrow \mathsf{EncK}(x)$; $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_N^{m_2}$;
    return $\mathbf{V} \leftarrow \begin{cases} g_2^{\mathbf{k}(0,\mathbf{r},\mathbf{h})} & \text{if } \mathfrak{d} = 0 \\ g_2^{\mathbf{k}(\alpha,\mathbf{r},\mathbf{h})} & \text{if } \mathfrak{d} = 1 \end{cases}$.

The advantage of adversary $\mathcal{A}$ against pair encoding P is defined as $\mathsf{Adv}^{t_1,t_2,\mathsf{G},P}_{\mathcal{A}}(\lambda) = |\Pr[\mathrm{Exp}_{P,\mathcal{G},\mathsf{G},0,\alpha,t_1,t_2}(\lambda) = 1] - \Pr[\mathrm{Exp}_{P,\mathcal{G},\mathsf{G},1,\alpha,t_1,t_2}(\lambda) = 1]|$. Pair encoding P is (t_1, t_2)-CMH (resp. SMH) secure in $\mathcal{G}$ if $\mathsf{Adv}^{t_1,t_2,\mathsf{G},P}_{\mathcal{A}}(\lambda)$ is negligible for $\mathsf{G} = \mathsf{CMH}$ (resp. SMH).

Remark 1. Similar to [Att16], we will use security notions like $(1, 1)$-CMH and $(1, \mathsf{poly})$-SMH security to prove the construction secure. In this paper, while proving security, we denote $(1, 1)$-CMH by CMH and $(1, \mathsf{poly})$-SMH by SMH security.

4 CCA-secure Predicate Encryption from Pair Encoding

Here we present a direct construction of CCA-secure predicate encryption scheme in prime-order groups from pair encoding scheme.

4.1 Regular Decryption Pair Encoding

We restrict underlying pair encoding to satisfy the property of *regular decryption*. These restrictions are amalgamation of regular encoding properties [Att16, Definition 1] and the decryption sufficiency properties [NP15b, Conditions 3.1]. Note that, these restrictions are quite natural and are observed in all the available pair encoding based predicate encryption construction.

The regular decryption properties of pair encoding is noted below:

$(\mathcal{P}1)$: For $\tilde{\iota} \in [w_1], \iota \in [m_1]$, if $\exists j' \in [0, w_2], k' \in [n], j \in [m_2], k \in [n]$ such that $a_{\tilde{\iota}j'k'} \neq 0$ and $b_{\iota jk} \neq 0$, then $\mathbf{E}_{\iota\tilde{\iota}} = 0$.
$(\mathcal{P}2)$: For $\iota \in [m_1]$, if $\exists j \in [m_2], k \in [n]$ such that $b_{\iota jk} \neq 0$ then $\exists \hat{\iota} \in [m_1]$ such that $k_{\hat{\iota}} = r_j$.
$(\mathcal{P}3)$: For $\tilde{\iota} \in [w_1]$, if $\exists j' \in [0, w_2], k' \in [n]$ such that $a_{\tilde{\iota}j'k'} \neq 0$ then $\exists \hat{\tilde{\iota}} \in [w_1]$ such that $c_{\hat{\tilde{\iota}}} = s_j$.
$(\mathcal{P}4)$: $c_1(\mathbf{s}, \mathbf{h}) = s_0$.
$(\mathcal{P}5)$: For $(x, y) \in \mathcal{X} \times \mathcal{Y}$, such that $R(x, y) = 1$, $(\mathbf{k}; m_2) \leftarrow \mathsf{EncK}(x, N)$ and $\mathbf{E} \leftarrow \mathsf{Pair}(x, y, N)$ then $\mathbf{k}(\alpha, \mathbf{0}, \mathbf{0})\mathbf{E} = (*, 0, \ldots, 0) \in \mathbb{Z}_N^{w_1}$ where $*$ is any non-zero entry.

Here we give some intuitive idea of regular decryption property of pair encoding. In Attrapadung's prime-order instantiation of pair encoding based predicate encryption schemes, a particular type of commutativity was impossible to compute [Att16, Eq. (8)]. We use Property $\mathcal{P}1$ to restrict such cases. This property has been used to prove the correctness of the scheme. Property $\mathcal{P}2$ and $\mathcal{P}3$ ensure that if the key-encoding $\mathbf{k}$ (resp. $\mathbf{c}$) contains $h_k r_j$ (resp. $h_{k'} s_{j'}$) then r_j (resp. $s_{j'}$) has to be be given explicitly. These two properties have been used in the security proof. We will see in the coming section that we produce a commitment on the *CPA-ciphertext* and bind it to the randomness s_0. Therefore we fix the position of polynomial s_0 in Property $\mathcal{P}4$. Also to decrypt, given a secret key $\mathbf{K} \in (\mathbb{G}_2^{(d+1)})^{m_1}$ and a pairing matrix $\mathbf{E} \in \mathbb{Z}_N^{m_1 \times w_1}$ (see Section 3 for description), the decryptor will compute an altKey $\hat{\mathbf{K}} = (\hat{\mathbf{K}}_0, \hat{\mathbf{K}}_1, \ldots, \hat{\mathbf{K}}_{w_1}) \in (\mathbb{G}_2^{(d+1)})^{(w_1+1)}$. We restrict that α used in secret key ($\mathbf{K}$) generation affects only $\hat{\mathbf{K}}_1$ via Property $\mathcal{P}5$. We will be needing this property in the security argument.

4.2 Construction

For a pair encoding scheme P for predicate function R, a predicate encryption Π_R for predicate function R is defined as following.

- $\mathsf{Setup}(1^\lambda, N)$: Runs $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p) \leftarrow \mathcal{G}(\lambda)$ where $\mathcal{G}$ is an asymmetric prime-order bilinear group generator. Picks $(g_1, g_2) \xleftarrow{\$} \mathbb{G}_1 \times \mathbb{G}_2$. Runs $n \leftarrow \mathsf{Param}(\kappa)$. Defines $\mathbb{H} = (\mathbf{H}_1, \ldots, \mathbf{H}_{n+2})$ where $\mathbf{H}_i \xleftarrow{\$} \mathbb{Z}_p^{(d+1)\times(d+1)}$ for each $i \in [n+2]$. Chooses $(\mathbf{B}, \tilde{\mathbf{D}}, \boldsymbol{\alpha}) \xleftarrow{\$} \mathbb{GL}_{p,d+1} \times \mathbb{GL}_{p,d} \times \mathbb{Z}_p^{(d+1)}$. Defines $\mathbf{D} := \left(\begin{smallmatrix} \tilde{\mathbf{D}} & \mathbf{0} \\ \mathbf{0} & 1 \end{smallmatrix}\right), \mathbf{Z} := \mathbf{B}^{-\top}\mathbf{D}$, chooses collision resistant hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p$ and chooses a one-time signature scheme OTS. Keeps $msk = (g_2^{\boldsymbol{\alpha}}, \mathbf{B}, \mathbf{Z}, \mathbb{H})$ to be secret and computes,
$$mpk = \left(g_1^{\mathbf{B}\left(\begin{smallmatrix}\mathbf{I}_d\\0\end{smallmatrix}\right)}, g_1^{\mathbf{H}_1\mathbf{B}\left(\begin{smallmatrix}\mathbf{I}_d\\0\end{smallmatrix}\right)}, \ldots, g_1^{\mathbf{H}_{n+2}\mathbf{B}\left(\begin{smallmatrix}\mathbf{I}_d\\0\end{smallmatrix}\right)}, g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathbf{I}_d\\0\end{smallmatrix}\right)}, g_2^{\mathbf{H}_1^\top\mathbf{Z}\left(\begin{smallmatrix}\mathbf{I}_d\\0\end{smallmatrix}\right)}, \ldots, g_2^{\mathbf{H}_{n+2}^\top\mathbf{Z}\left(\begin{smallmatrix}\mathbf{I}_d\\0\end{smallmatrix}\right)}, e(g_1,g_2)^{\boldsymbol{\alpha}^\top\mathbf{B}\left(\begin{smallmatrix}\mathbf{I}_d\\0\end{smallmatrix}\right)}, \mathcal{H}, \mathsf{OTS} \right).$$
- $\mathsf{KeyGen}(msk, x)$: Runs $(\mathbf{k} = (k_1, \ldots, k_{m_1}); m_2) \leftarrow \mathsf{EncK}(x, N)$. Chooses $\mathbf{r}_1, \ldots, \mathbf{r}_{m_2} \xleftarrow{\$} \mathbb{Z}_p^d$ and defines $\mathbf{R} = \left(\left(\begin{smallmatrix}\mathbf{r}_1\\0\end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix}\mathbf{r}_{m_2}\\0\end{smallmatrix}\right)\right) \in \left(\mathbb{Z}_p^{(d+1)}\right)^{m_2}$. Outputs $\mathbf{K} = \{g_2^{k_\iota(\boldsymbol{\alpha},\mathbf{R},\mathbb{H})}\}_{\iota\in[m_1]} \in \left(\mathbb{G}_2^{(d+1)}\right)^{m_1}$ where for each $\iota \in [m_1]$,
$$k_\iota(\boldsymbol{\alpha}, \mathbf{R}, \mathbb{H}) = b_\iota\boldsymbol{\alpha} + \sum_{j\in[m_2]} b_{\iota j}\mathbf{Z}\left(\begin{smallmatrix}\mathbf{r}_j\\0\end{smallmatrix}\right) + \sum_{\substack{j\in[m_2]\\k\in[n]}} b_{\iota jk}\mathbf{H}_k^\top\mathbf{Z}\left(\begin{smallmatrix}\mathbf{r}_j\\0\end{smallmatrix}\right).$$
- $\mathsf{Enc}(mpk, y, M)$: Runs $(\mathbf{c} = (c_1, \ldots, c_{w_1}); w_2) \leftarrow \mathsf{EncC}(y, N)$. Chooses $\mathbf{s}_0, \ldots, \mathbf{s}_{w_2} \xleftarrow{\$} \mathbb{Z}_p^d$ and defines $\mathbf{S} = \left(\left(\begin{smallmatrix}\mathbf{s}_0\\0\end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix}\mathbf{s}_{w_2}\\0\end{smallmatrix}\right)\right) \in \left(\mathbb{Z}_p^{(d+1)}\right)^{(w_2+1)}$. Computes $\mathbf{C} = (\mathbf{C}_1, \ldots, \mathbf{C}_{w_1}, \mathbf{C}_{w_1+1})$ where for each $\tilde{\iota} \in [w_1]$, $\mathbf{C}_{\tilde{\iota}} = g_1^{c_{\tilde{\iota}}(\mathbf{S},\mathbb{H})} \in \mathbb{G}_1^{(d+1)}$ such that
$$c_{\tilde{\iota}}(\mathbf{S}, \mathbb{H}) = \sum_{j\in[0,w_2]} a_{\tilde{\iota}j}\mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_j\\0\end{smallmatrix}\right) + \sum_{\substack{j\in[0,w_2]\\k\in[n]}} a_{\tilde{\iota}jk}\mathbf{H}_k\mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_j\\0\end{smallmatrix}\right) \text{ for } \tilde{\iota} \in [w_1]$$
and $\mathbf{C}_{w_1+1} = M \cdot e(g_1, g_2)^{\boldsymbol{\alpha}^\top\mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\0\end{smallmatrix}\right)}$. Runs $(vk, sk) \leftarrow \mathsf{OTS.Gen}(1^\lambda)$. Then it computes $\xi = \mathcal{H}(\mathbf{C}, vk)$ and outputs $\overline{\mathbf{C}} = (\overline{\mathbf{C}}_0, \mathbf{C}, vk, \sigma)$ where $\overline{\mathbf{C}}_0 = g_1^{(\xi\mathbf{H}_{n+1}+\mathbf{H}_{n+2})\mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\0\end{smallmatrix}\right)}$ and $\sigma \leftarrow \mathsf{OTS.Sign}(sk, \overline{\mathbf{C}}_0)$.
- $\mathsf{Dec}(\mathbf{K}, \overline{\mathbf{C}})$: Given $\mathbf{K}$ and $\overline{\mathbf{C}}$ corresponding to key-index x and data-index y respectively, if $R(x, y) = 0$, it aborts. Also aborts if $\mathsf{OTS.Verify}(\overline{\mathbf{C}}_0, vk, \sigma)$ evaluates to 0. Then runs $\mathbf{E} \leftarrow \mathsf{Pair}(x, y, N)$. Given $\mathbf{K} = (\mathbf{K}_1, \ldots, \mathbf{K}_{m_1})$ and ciphertext $\overline{\mathbf{C}}$ it computes $(\tilde{K}_1, \ldots, \tilde{K}_{w_1})$ where $\tilde{K}_{\tilde{\iota}} = \prod_{\iota\in[m_1]} (\mathbf{K}_\iota)^{\mathbf{E}_{\iota\tilde{\iota}}}$ for each $\tilde{\iota} \in [w_1]$. Chooses $\mathfrak{r} \xleftarrow{\$} \mathbb{Z}_p^d$. Defines modified key $\hat{\mathbf{K}} = (K_0, \Phi \cdot \tilde{K}_1, \tilde{K}_2, \ldots, \tilde{K}_{w_1})$ where $K_0 = g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\0\end{smallmatrix}\right)}$ and $\Phi = g_2^{(\xi\mathbf{H}_{n+1}^\top+\mathbf{H}_{n+2}^\top)\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\0\end{smallmatrix}\right)}$ for $\xi = \mathcal{H}(\mathbf{C}, vk)$. Outputs M such that
$$M = \mathbf{C}_{w_1+1} \cdot e(\overline{\mathbf{C}}_0, K_0) \cdot \left(\prod_{\tilde{\iota}\in[w_1]} e(\mathbf{C}_{\tilde{\iota}}, \hat{\mathbf{K}}_{\tilde{\iota}}) \right)^{-1}. \tag{1}$$

4.3 Correctness

Here we compute $e(\overline{\mathbf{C}}_0, K_0) \cdot \left(\prod_{\tilde{\iota}\in[w_1]} e(\mathbf{C}_{\tilde{\iota}}, \hat{\mathbf{K}}_{\tilde{\iota}}) \right)^{-1}$ to show the correctness of the construction.

$$e(\overline{\mathbf{C}}_0, K_0) = e\left(g_1^{(\xi\mathbf{H}_{n+1}+\mathbf{H}_{n+2})\mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\0\end{smallmatrix}\right)}, g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\0\end{smallmatrix}\right)} \right) = e\left(g_1^{\mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\0\end{smallmatrix}\right)}, g_2^{(\xi\mathbf{H}_{n+1}^\top+\mathbf{H}_{n+2}^\top)\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\0\end{smallmatrix}\right)} \right).$$

Now, $\prod_{\tilde{\iota}\in[w_1]} e(\mathbf{C}_{\tilde{\iota}}, \hat{\mathbf{K}}_{\tilde{\iota}})$

$$= e(\mathbf{C}_1, \hat{\mathbf{K}}_1) \cdot \prod_{\tilde{\iota}\in[2,w_1]} e(\mathbf{C}_{\tilde{\iota}}, \hat{\mathbf{K}}_{\tilde{\iota}})$$

$$= e\left(g_1^{\mathbf{B}\begin{pmatrix}\mathbf{s}_0\\0\end{pmatrix}}, \Phi \cdot \tilde{K}_1\right) \cdot \prod_{\tilde{\iota}\in[2,w_1]} e(\mathbf{C}_{\tilde{\iota}}, \hat{\mathbf{K}}_{\tilde{\iota}}) \qquad \text{(due to Property } \mathcal{P}4 \text{ of regular decryption pair encoding)}$$

$$= e\left(g_1^{\mathbf{B}\begin{pmatrix}\mathbf{s}_0\\0\end{pmatrix}}, \Phi\right) \cdot e\left(g_1^{\mathbf{B}\begin{pmatrix}\mathbf{s}_0\\0\end{pmatrix}}, \tilde{K}_1\right) \cdot \prod_{\tilde{\iota}\in[2,w_1]} e(\mathbf{C}_{\tilde{\iota}}, \tilde{K}_{\tilde{\iota}})$$

$$= e\left(g_1^{\mathbf{B}\begin{pmatrix}\mathbf{s}_0\\0\end{pmatrix}}, g_2^{(\xi\mathbf{H}_{n+1}^\top + \mathbf{H}_{n+2}^\top)\mathbf{Z}\begin{pmatrix}\mathfrak{r}\\0\end{pmatrix}}\right) \cdot e(\mathbf{C}_1, \tilde{K}_1) \cdot \prod_{\tilde{\iota}\in[2,w_1]} e(\mathbf{C}_{\tilde{\iota}}, \tilde{K}_{\tilde{\iota}})$$

$$= e\left(g_1^{\mathbf{B}\begin{pmatrix}\mathbf{s}_0\\0\end{pmatrix}}, g_2^{(\xi\mathbf{H}_{n+1}^\top + \mathbf{H}_{n+2}^\top)\mathbf{Z}\begin{pmatrix}\mathfrak{r}\\0\end{pmatrix}}\right) \cdot \prod_{\tilde{\iota}\in[w_1]} e(\mathbf{C}_{\tilde{\iota}}, \tilde{K}_{\tilde{\iota}})$$

Then $e(\overline{\mathbf{C}}_0, K_0) \cdot \left(\prod_{\tilde{\iota}\in[w_1]} e(\mathbf{C}_{\tilde{\iota}}, \hat{\mathbf{K}}_{\tilde{\iota}})\right)^{-1}$

$$= e\left(g_1^{\mathbf{B}\begin{pmatrix}\mathbf{s}_0\\0\end{pmatrix}}, g_2^{(\xi\mathbf{H}_{n+1}^\top + \mathbf{H}_{n+2}^\top)\mathbf{Z}\begin{pmatrix}\mathfrak{r}\\0\end{pmatrix}}\right) \cdot e\left(g_1^{\mathbf{B}\begin{pmatrix}\mathbf{s}_0\\0\end{pmatrix}}, g_2^{(\xi\mathbf{H}_{n+1}^\top + \mathbf{H}_{n+2}^\top)\mathbf{Z}\begin{pmatrix}\mathfrak{r}\\0\end{pmatrix}}\right)^{-1} \cdot \left(\prod_{\tilde{\iota}\in[w_1]} e(\mathbf{C}_{\tilde{\iota}}, \tilde{K}_{\tilde{\iota}})\right)^{-1}$$

$$= \left(\prod_{\tilde{\iota}\in[w_1]} e(\mathbf{C}_{\tilde{\iota}}, \prod_{\iota\in[m_1]} (\mathbf{K}_\iota)^{\mathbf{E}_{\iota\tilde{\iota}}})\right)^{-1}$$

$$= \left(\prod_{\substack{\iota\in[m_1]\\ \tilde{\iota}\in[w_1]}} e(\mathbf{C}_{\tilde{\iota}}, \mathbf{K}_\iota)^{\mathbf{E}_{\iota\tilde{\iota}}}\right)^{-1}$$

$$= e(g_1, g_2)^{-\boldsymbol{\alpha}^\top \mathbf{B}\begin{pmatrix}\mathbf{s}_0\\0\end{pmatrix}} \qquad \text{(due to correctness of [Att16])}$$

Then $\mathbf{C}_{w_1+1} \cdot e(\overline{\mathbf{C}}_0, K_0) \cdot \left(\prod_{\tilde{\iota}\in[w_1]} e(\mathbf{C}_{\tilde{\iota}}, \hat{\mathbf{K}}_{\tilde{\iota}})\right)^{-1} = M.$

Remark 2. Dec creates *modified key* $\hat{\mathbf{K}}$ for a given secret key $\mathbf{K}$ and the pairing matrix $\mathbf{E}$. From now onwards, we will use *decryption key* or *altKey* interchangeably to denote the *modified key*. One can simply view the Dec function as a composition of $\mathsf{AltKeyGen}$ and AltDec where the former creates the *altKey* $\hat{\mathbf{K}}$ and the latter performs the decryption of $\overline{\mathbf{C}}$ using *altKey* $\hat{\mathbf{K}}$. Precisely $\mathsf{AltKeyGen}(\overline{\mathbf{C}}, x, msk)$ computes modified key $\hat{\mathbf{K}}$ and $\mathsf{AltDec}(\overline{\mathbf{C}}, \hat{\mathbf{K}})$ computes RHS of Eq. (1).

Remark 3. We have extended the CPA-secure predicate encryption construction of [Att16]. We follow the direct CCA construction technique of [NP15b] to achieve an efficient CCA-secure predicate encryption construction in prime order groups. Note that [NP15b] was instantiated in composite order groups. The common variables $\mathbb{H}$, in our construction, contains $n+2$ matrices whereas in [Att16] it contained n matrices. We use these extra two common variables $\mathbf{H}_{n+1}$ and $\mathbf{H}_{n+2}$ to compute a commitment of CPA-ciphertext $\mathbf{C}$. This technique loosely relates to [BMW05]. We first compute the hash of $(\mathbf{C}, vk)$ and bind it to the randomness $\mathbf{B}\begin{pmatrix}\mathbf{s}_0\\0\end{pmatrix}$ using common variables $\mathbf{H}_{n+1}$ and $\mathbf{H}_{n+2}$ where vk is verification key for OTS. This results in an extra ciphertext component namely $\overline{\mathbf{C}}_0$. We then use the one-time signature OTS to compute a signature σ on $\overline{\mathbf{C}}_0$ and output $\overline{\mathbf{C}} = (\overline{\mathbf{C}}_0, \mathbf{C}, vk, \sigma)$. Notice that KeyGen algorithm is exactly the same as [Att16]. The Dec is modified to perform cancellation of extra ciphertext component. To do that, we define altKey $\hat{\mathbf{K}}$ to contain K_0 and Φ. We use *associativity* [Att16, Section 4.1] to cancel the extra ciphertext component $\overline{\mathbf{C}}_0$ using K_0 and Φ. Once such a cancellation is performed, the decryption happens exactly like [Att16].

Remark 4. Even if our construction here is structurally quite similar to that of [NP15b], we use an OTS to ensure *integrity* of $\overline{\mathbf{C}}_0$. Use of OTS allows us to get rid of extra verification step (precisely Eq. (7) in Appendix C) that is needed to check the structure of $\overline{\mathbf{C}}_0$. We have presented the construction of direct CCA-secure predicate encryption in prime order groups in Appendix C, that performs such a check and therefore follows the construction of [NP15b] completely.

Efficiency. Our construction increases the ciphertext length by exactly three namely $\overline{\mathbf{C}}_0$, vk and σ is returned along with the CPA-ciphertext $\mathbf{C}$ where $\overline{\mathbf{C}}_0 \in \mathbb{G}_1^{(d+1)}$, vk is verification key of OTS and σ is the signature of $\overline{\mathbf{C}}_0$ with respect to the signing key sk corresponding to vk. As we mentioned earlier, we have reused KeyGen of [Att16], therefore the secret key does not change. However Dec has to verify the signature σ and evaluate only one additional unit of pairing (namely $e(\overline{\mathbf{C}}_0, K_0)$). As both $\overline{\mathbf{C}}_0$ and K_0 are group elements having $(d+1)$-components, the decryption in our construction incurs an additional cost of $(d+1)$ pairing evaluations only. This is really efficient as opposed to traditional CPA to CCA conversions by [YAHK11,YAS+12,NP17] that needs almost two times $m_1 \times w_1 \times (d+1) \times (m_2+1) \times d$ many pairing evaluations. See Appendix A for the exact cost of such conversions.

5 Security of the Proposed Construction

To prove our predicate encryption construction fully CCA-secure, we extend the proof technique used by Attrapadung in [Att16]. In case of dual system proof technique, one needs to add randomness to ciphertext, keys and altKeys to construct semi-functional ciphertext, semi-functional key and semi-functional altKeys respectively. At the end, one has to show that the randomness of semi-functional components of ciphertext and keys will blind the message completely. We use the abbreviation 'type' to identify semi-functional type.

Intuitively, the collision resistance of $\mathcal{H}$ neither allows the adversary to come up with a different $\mathbf{C}$ nor allows the adversary to change vk that results in the same *commitment* ξ. The adversary, after receiving challenge $\overline{\mathbf{C}}^* = (\overline{\mathbf{C}}_0^*, \mathbf{C}^*, vk^*, \sigma^*)$, can however keep the same $\mathbf{C}^*$ and construct some different $\overline{\mathbf{C}}_0'$ and produce $\overline{\mathbf{C}} = (\overline{\mathbf{C}}_0', \mathbf{C}^*, vk^*, \sigma^*)$ as decryption query. Such a scenario allows the simulator to forge the underlying one-time signature. Therefore during the security game, what the adversary can do is to come up with random ciphertext $\overline{\mathbf{C}}$ for decryption. With all but negligible probability, x used in decryption query $(x, \overline{\mathbf{C}})$ will not satisfy y which is implicit data-index of $\overline{\mathbf{C}}$. This way we are ultimately stopping the adversary to gather any non-trivial information.

Theorem 1. *Suppose a regular decryption pair encoding scheme P for predicate R is both* SMH *and* CMH*-secure*[2] *in $\mathcal{G}$, and the $\mathcal{D}_d$-Matrix DH Assumption holds in $\mathcal{G}$. Then the scheme Π_R (in Section 4.2) is fully CCA-secure encryption scheme if $\mathcal{H}$ is collision resistant hash function and OTS is strong one-time signature. More precisely, for any PPT adversary $\mathcal{A}$ that makes at most q_1 key queries before challenge, at most q_2 key queries after challenge and at most $q_{_D}$ decryption queries throughout the game, there exists PPT algorithms $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4, \mathcal{B}_5$ such that for any λ,*

$$\mathsf{Adv}^{\mathrm{PE}}_{\mathcal{A}}(\lambda) \leq (2q_1 + 2q_{_D} + 3) \cdot \mathsf{Adv}^{\mathcal{D}_d\text{-MatDH}}_{\mathcal{B}_1}(\lambda) + q_1 \cdot \mathsf{Adv}^{\mathrm{CMH}}_{\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathrm{SMH}}_{\mathcal{B}_3}(\lambda) + q_{_D} \cdot \mathsf{Adv}^{\mathrm{CRH}}_{\mathcal{B}_4}(\lambda) + q_{_D} \cdot \mathsf{Adv}^{\mathrm{sUf\text{-}CMA}}_{\mathcal{B}_5, \mathsf{OTS}}(\lambda).$$

5.1 Security Argument

Here we give hybrid security argument to prove the security of predicate encryption scheme Π_R. Probabilistic polynomial time adversary $\mathcal{A}$ is capable of making at most q_1 key queries before challenge phase, at most q_2 key queries after challenge phase and at most $q_{_D}$ decryption queries throughout the game. Let $q = q_1 + q_2$.

Game_0 is the real security game and Game_4 is the game where all secret keys are type-3 semi-functional keys, all altKeys are type-3 semi-functional altKeys and the challenge ciphertext is semi-functional ciphertext of random message (therefore is independent of the message that is to be encrypted). To prove the indistinguishability of Game_0 and Game_4, we define the sequence of games of Table 1. The idea is to change each game only by a small margin and prove indistinguishability of two consecutive games. First we make the challenge ciphertext semi-functional. Then

[2] Here SMH means $(1, \mathsf{poly})$-SMH and CMH means $(1,1)$-CMH (See Section 3.1).

Games	Difference from Previous	Proof Strategy
Game_0	-	[Att16]
Game_1	challenge ciphertext is semi-functional	[Att16]
$\mathsf{Game}_{2,i-1,3}$	all the $(i-1)$ secret keys are type-3 key $(i \leq q_1)$	[Att16]
$\mathsf{Game}_{2,i,1}$	i^{th} secret key is type-1 key $(i \leq q_1)$	[Att16]
$\mathsf{Game}_{2,i,2}$	i^{th} secret key is type-2 key $(i \leq q_1)$	[Att16]
$\mathsf{Game}_{2,i,3}$	i^{th} secret key is type-3 key $(i \leq q_1)$	[Att16]
$\mathsf{Game}_{2,q_1+1,1}$	all post-challenge secret keys are type-1 key	[Att16]
$\mathsf{Game}_{2,q_1+1,2}$	all post-challenge secret keys are type-2 key	[Att16]
$\mathsf{Game}_{2,q_1+1,3}$	all post-challenge secret keys are type-3 key	[Att16]
$\mathsf{Game}_{3,i-1,3}$	all the $(i-1)$ altKeys are type-3 altKey $(i \leq q_D)$	this work
$\mathsf{Game}_{3,i,1}$	i^{th} altKey is type-1 altKey $(i \leq q_D)$	this work
$\mathsf{Game}_{3,i,2}$	i^{th} altKey is type-2 altKey $(i \leq q_D)$	this work
$\mathsf{Game}_{3,i,3}$	i^{th} altKey is type-3 altKey $(i \leq q_D)$	this work
Game_4	challenge ciphertext is semi-functional encryption of a random message	this work

Table 1. Outline of proof strategy

we modify each i^{th} pre-challenge key to type-j semi-functional key in $\mathsf{Game}_{2,i,j}$ for each $i \in [q_1]$ and $j \in \{1,2,3\}$. Note that to answer i^{th} pre-challenge key query, the simulator chooses fresh $\beta_i \xleftarrow{\$} \mathbb{Z}_p$. Then we modify all the post-challenge keys to type-j keys together in $\mathsf{Game}_{2,q_1+1,j}$ for each $i \in [q_1+1, q]$ and $j \in \{1,2,3\}$. Here however the simulator uses same $\beta \xleftarrow{\$} \mathbb{Z}_p$ to answer every post-challenge key query. Then we modify each (i^{th}) altKey to type-j semi-functional altKey in $\mathsf{Game}_{3,i,j}$ for each $i \in [q_D]$ and $j \in \{1,2,3\}$. Note that the simulator uses same $\eta \xleftarrow{\$} \mathbb{Z}_p$ to compute all the altKeys. In the final game Game_4, we show that the ciphertext is completely independent of the message it is encrypting. Therefore the advantage of adversary $\mathcal{A}$ in Game_4 is 0. Note that Game_1 and $\mathsf{Game}_{2,q_1+1,3}$ are also denoted by $\mathsf{Game}_{2,0,3}$ and $\mathsf{Game}_{3,0,3}$ respectively.

As mentioned in Table 1, we have used the proof technique of [Att16] to argue indistinguishability of several games. However, the games that *deal with* changes in altKey and the final game are our contribution. We have put the description of games that we have mimicked from [Att16] in Appendix B. Here we concentrate only in $\mathsf{Game}_{3,i,1}$, $\mathsf{Game}_{3,i,2}$, $\mathsf{Game}_{3,i,3}$ for $i \in [q_D]$ and Game_4.

Note that in $\mathsf{Game}_{2,q_1+1,3}$, the challenge ciphertext is semi-functional and all the secret keys are type-3 semi-functional. However, all the altKeys at this moment are normal. We then change the altKeys to type-3 semi-functional altKey one by one. For every $i \in [q_D]$, this is done via changing normal altKey to type-1 altKey first in $\mathsf{Game}_{3,i,1}$. Subsequently we change it into type-2 altKey in $\mathsf{Game}_{3,i,2}$ and to type-3 altKey in $\mathsf{Game}_{3,i,3}$. In $\mathsf{Game}_{3,i,2}$ we introduce the randomness η that hides the master secret key in the final game. This effectively allows us to show that in the final game, the simulator can simulate all the secret keys and the altKeys properly and the challenge ciphertext is semi-functional ciphertext of random message.

5.2 Semi-functional Algorithms

Following semi-functional algorithms will be used in the security proof.

- $\mathsf{SFSetup}(1^\lambda, \kappa)$: It runs $(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda, \kappa)$. Additionally it outputs $\widehat{mpk}_{base}$, $\widehat{mpk}_b$ and $\widehat{mpk}_z$ where $\widehat{mpk}_{base} = g_2^{\mathbf{Z}\binom{\mathbf{0}}{1}}$, $\widehat{mpk}_b = \left(e(g_1,g_2)^{\boldsymbol{\alpha}^\top \mathbf{B}\binom{\mathbf{0}}{1}}, g_1^{\mathbf{B}\binom{\mathbf{0}}{1}}, g_1^{\mathbf{H}_1\mathbf{B}\binom{\mathbf{0}}{1}}, \ldots, g_1^{\mathbf{H}_{n+2}\mathbf{B}\binom{\mathbf{0}}{1}}\right)$ and $\widehat{mpk}_z = \left(g_2^{\mathbf{H}_1^\top \mathbf{Z}\binom{\mathbf{0}}{1}}, \ldots, g_2^{\mathbf{H}_{n+2}^\top \mathbf{Z}\binom{\mathbf{0}}{1}}\right)$.
- $\mathsf{SFKeyGen}(x, msk, \widehat{mpk}_z, \widehat{mpk}_{base}, \text{type}, \beta)$: Runs $(\mathbf{k}; m_2) \leftarrow \mathsf{EncK}(x, N)$. Chooses $\mathbf{r}_1, \ldots, \mathbf{r}_{m_2} \xleftarrow{\$} \mathbb{Z}_p^d$ and $\hat{r}_1, \ldots, \hat{r}_{m_2} \xleftarrow{\$} \mathbb{Z}_p$. Then it defines $\mathbf{R} = \left(\binom{\mathbf{r}_1}{0}, \ldots, \binom{\mathbf{r}_{m_2}}{0}\right) \in \left(\mathbb{Z}_p^{(d+1)}\right)^{m_2}$ and $\widehat{\mathbf{R}} = \left(\binom{\mathbf{0}}{\hat{r}_1}, \ldots, \binom{\mathbf{0}}{\hat{r}_{m_2}}\right) \in \left(\mathbb{Z}_p^{(d+1)}\right)^{m_2}$. Outputs the secret key

$$\mathbf{K} = \begin{cases} g_2^{\mathbf{k}(\boldsymbol{\alpha},\mathbf{R},\mathbb{H})+\mathbf{k}(\mathbf{0},\widehat{\mathbf{R}},\mathbb{H})} & \text{if type} = 1 \\ g_2^{\mathbf{k}(\boldsymbol{\alpha},\mathbf{R},\mathbb{H})+\mathbf{k}(\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ \beta\end{smallmatrix}\right),\widehat{\mathbf{R}},\mathbb{H})} & \text{if type} = 2 \\ g_2^{\mathbf{k}(\boldsymbol{\alpha},\mathbf{R},\mathbb{H})+\mathbf{k}(\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ \beta\end{smallmatrix}\right),\mathbf{0},\mathbb{H})} & \text{if type} = 3 \end{cases}$$

where $\mathbf{k}(\boldsymbol{\alpha},\mathbf{R},\mathbb{H}) + \mathbf{k}(\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ \beta\end{smallmatrix}\right),\widehat{\mathbf{R}},\mathbb{H}) =$

$$\left\{ b_\iota \boldsymbol{\alpha} + b_\iota \mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ \beta\end{smallmatrix}\right) + \sum_{j\in[m_2]} b_{\iota j}\mathbf{Z}\left(\begin{smallmatrix}\mathbf{r}_j\\ \hat{r}_j\end{smallmatrix}\right) + \sum_{\substack{j\in[m_2]\\ k\in[n]}} b_{\iota jk}\mathbf{H}_k^\top \mathbf{Z}\left(\begin{smallmatrix}\mathbf{r}_j\\ \hat{r}_j\end{smallmatrix}\right) \right\}_{\iota\in[m_1]}.$$

– $\mathsf{SFEnc}(y, M, mpk, \widehat{mpk}_b)$: It runs $(\mathbf{c}; w_2) \leftarrow \mathsf{EncC}(y, N)$. Chooses $\mathbf{s}_0, \ldots, \mathbf{s}_{w_2} \xleftarrow{\$} \mathbb{Z}_p^d$ and $\hat{s}_0, \ldots, \hat{s}_{w_2} \xleftarrow{\$} \mathbb{Z}_p$. It defines $\mathbf{S} = \left(\left(\begin{smallmatrix}\mathbf{s}_0\\ 0\end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix}\mathbf{s}_{w_2}\\ 0\end{smallmatrix}\right)\right) \in \left(\mathbb{Z}_p^{(d+1)}\right)^{(w_2+1)}$ and $\widehat{\mathbf{S}} = \left(\left(\begin{smallmatrix}\mathbf{0}\\ \hat{s}_0\end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix}\mathbf{0}\\ \hat{s}_{w_2}\end{smallmatrix}\right)\right) \in \left(\mathbb{Z}_p^{(d+1)}\right)^{(w_2+1)}$. It computes the semi-functional ciphertext $\mathbf{C} = (\mathbf{C}_1, \ldots, \mathbf{C}_{w_1}, \mathbf{C}_{w_1+1})$ where

$$\mathbf{C}_{\tilde{\iota}} = g_1^{\mathbf{c}_{\tilde{\iota}}(\mathbf{S},\mathbb{H})+\mathbf{c}_{\tilde{\iota}}(\widehat{\mathbf{S}},\mathbb{H})} = g_1^{\left(\sum_{j\in[0,w_2]} a_{\tilde{\iota}j}\mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_j\\ \hat{s}_j\end{smallmatrix}\right) + \sum_{\substack{j\in[0,w_2]\\ k\in[n]}} a_{\tilde{\iota}jk}\mathbf{H}_k\mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_j\\ \hat{s}_j\end{smallmatrix}\right)\right)}$$

for $\tilde{\iota} \in [w_1]$ and $\mathbf{C}_{w_1+1} = M \cdot e(g_1, g_2)^{\boldsymbol{\alpha}^\top \mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\ \hat{s}_0\end{smallmatrix}\right)}$. Runs $(vk, sk) \leftarrow \mathsf{OTS.Gen}(1^\lambda)$. Then it computes $\xi = \mathcal{H}(\mathbf{C}, vk)$ and outputs $\overline{\mathbf{C}} = (\overline{\mathbf{C}}_0, \mathbf{C}, vk, \sigma)$ where $\overline{\mathbf{C}}_0 = g_1^{(\xi\mathbf{H}_{n+1}+\mathbf{H}_{n+2})\mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\ \hat{s}_0\end{smallmatrix}\right)}$ and $\sigma \leftarrow \mathsf{OTS.Sign}(sk, \overline{\mathbf{C}}_0)$.

– $\mathsf{SFAltKeyGen}(\overline{\mathbf{C}}, x, msk, \widehat{mpk}_z, \widehat{mpk}_{base}, \text{type}, \eta)$: Runs $(\mathbf{k}; m_2) \leftarrow \mathsf{EncK}(x, N)$ and $\mathbf{E} \leftarrow \mathsf{Pair}(x, y, N)$. Chooses $\mathbf{r}_1, \ldots, \mathbf{r}_{m_2}, \mathfrak{r} \xleftarrow{\$} \mathbb{Z}_p^d$ and $\hat{\mathfrak{r}} \xleftarrow{\$} \mathbb{Z}_p$. Then it defines $\mathbf{R} = \left(\left(\begin{smallmatrix}\mathbf{r}_1\\ 0\end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix}\mathbf{r}_{m_2}\\ 0\end{smallmatrix}\right)\right) \in \left(\mathbb{Z}_p^{(d+1)}\right)^{m_2}$.

Then the normal key is $\mathbf{K} = \left\{g_2^{k_\iota(\boldsymbol{\alpha},\mathbf{R},\mathbb{H})}\right\}_{\iota\in[m_1]} \in \left(\mathbb{G}_2^{(d+1)}\right)^{m_1}$ where

$$k_\iota(\boldsymbol{\alpha},\mathbf{R},\mathbb{H}) = b_\iota\boldsymbol{\alpha} + \sum_{j\in[m_2]} b_{\iota j}\mathbf{Z}\left(\begin{smallmatrix}\mathbf{r}_j\\ 0\end{smallmatrix}\right) + \sum_{\substack{j\in[m_2]\\ k\in[n]}} b_{\iota jk}\mathbf{H}_k^\top\mathbf{Z}\left(\begin{smallmatrix}\mathbf{r}_j\\ 0\end{smallmatrix}\right) \text{ for } \iota \in [m_1].$$

Then it computes $(\tilde{K}_1, \ldots, \tilde{K}_{w_1})$ where $\tilde{K}_{\tilde{\iota}} = \prod_{\iota\in[m_1]} (\mathbf{K}_\iota)^{\mathbf{E}_{\iota\tilde{\iota}}}$ for $\tilde{\iota} \in [w_1]$.

Defines modified key $\hat{\mathbf{K}} = (K_0, \Phi \cdot \tilde{K}_1, \tilde{K}_2, \ldots, \tilde{K}_{w_1})$ where

$$(K_0, \Phi) = \begin{cases} \left(g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{smallmatrix}\right)}, g_2^{(\xi\mathbf{H}_{n+1}^\top+\mathbf{H}_{n+2}^\top)\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{smallmatrix}\right)}\right) & \text{if type} = 1 \\ \left(g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{smallmatrix}\right)}, g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ \eta u\end{smallmatrix}\right)+(\xi\mathbf{H}_{n+1}^\top+\mathbf{H}_{n+2}^\top)\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{smallmatrix}\right)}\right) & \text{if type} = 2 \\ \left(g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\ 0\end{smallmatrix}\right)}, g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ \eta u\end{smallmatrix}\right)+(\xi\mathbf{H}_{n+1}^\top+\mathbf{H}_{n+2}^\top)\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\ 0\end{smallmatrix}\right)}\right) & \text{if type} = 3 \end{cases}$$

for $u = \sum_{\iota\in[m_1]} b_\iota \mathbf{E}_{\iota 1}$ and $\xi = \mathcal{H}(\mathbf{C}, vk)$ in case of given ciphertext $\overline{\mathbf{C}} = (\overline{\mathbf{C}}_0, \mathbf{C}, vk, \sigma)$.

5.3 Normal to Type-1 altKey

Lemma 1 ($\mathsf{Game}_{3,i-1,3}$ **to** $\mathsf{Game}_{3,i,1}$)**.** *For $i = 1, \ldots, q_{_D}$, for any efficient adversary $\mathcal{A}$ that makes at most q key queries and at most $q_{_D}$ decryption queries, there exists a PPT algorithm $\mathcal{B}$ such that $|\mathsf{Adv}_{\mathcal{A}}^{3,i-1,3}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{3,i,1}(\lambda)| \leq \mathsf{Adv}_{\mathcal{B}}^{\mathcal{D}_d\text{-MatDH}}(\lambda)$.*

Proof. The algorithm $\mathcal{B}$ gets input $(\mathbf{G}, g_2^{\mathbf{T}}, g_2^{\mathbf{T}\begin{pmatrix}\boldsymbol{y}\\ \hat{y}\end{pmatrix}})$ as $\mathcal{D}_d$-MatDH problem instance where $\hat{y} = 0$ or $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$, $\boldsymbol{y} \xleftarrow{\$} \mathbb{Z}_p^d$.

Setup. $\mathcal{B}$ chooses $\tilde{\mathbf{B}} \xleftarrow{\$} \mathbb{GL}_{p,d+1}, \mathbf{J} \xleftarrow{\$} \mathbb{GL}_{p,d}$ and sets $\mathbf{B} = \tilde{\mathbf{B}}\begin{pmatrix}\mathbf{I}_d & \mathbf{M}^{-\top}\mathbf{c}^\top\\ \mathbf{0} & -1\end{pmatrix}$ and $\mathbf{D} = \begin{pmatrix}\mathbf{MJ} & \mathbf{0}\\ \mathbf{0} & 1\end{pmatrix}$ where $\mathbf{T} = \begin{pmatrix}\mathbf{M} & \mathbf{0}\\ \mathbf{c} & 1\end{pmatrix}$ due to $\mathcal{D}_d$-MatDH assumption. Then $\mathbf{Z} = \mathbf{B}^{-\top}\mathbf{D} = \tilde{\mathbf{B}}^{-\top}\begin{pmatrix}\mathbf{I}_d & \mathbf{0}\\ \mathbf{cM}^{-1} & -1\end{pmatrix}\begin{pmatrix}\mathbf{MJ} & \mathbf{0}\\ \mathbf{0} & 1\end{pmatrix} = \tilde{\mathbf{B}}^{-\top}\mathbf{T}\begin{pmatrix}\mathbf{J} & \mathbf{0}\\ \mathbf{0} & -1\end{pmatrix}$. Then define $\tilde{\mathbf{Z}} = \begin{pmatrix}\mathbf{J} & \mathbf{0}\\ \mathbf{0} & -1\end{pmatrix}$ so that $\mathbf{Z} = \tilde{\mathbf{B}}^{-\top}\mathbf{T}\tilde{\mathbf{Z}}$. $\mathcal{B}$ therefore can compute the public parameters as $g_1^{\mathbf{B}\begin{pmatrix}\mathbf{I}_d\\ 0\end{pmatrix}} = g_1^{\tilde{\mathbf{B}}\begin{pmatrix}\mathbf{I}_d\\ 0\end{pmatrix}}$ and $g_2^{\mathbf{Z}} = g_2^{\tilde{\mathbf{B}}^{-\top}\mathbf{T}\tilde{\mathbf{Z}}}$. Then $\mathcal{B}$ chooses $\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)}$ and $\mathbb{H} = (\mathbf{H}_1, \ldots, \mathbf{H}_{n+2}) \xleftarrow{\$} \left(\mathbb{Z}_p^{(d+1)\times(d+1)}\right)^{(n+2)}$ and publishes public key mpk. Note that $\mathcal{B}$ cannot compute $\widehat{mpk}_b$ but can compute $\widehat{mpk}_z$ as it can compute $\widehat{mpk}_{base}$. It chooses $\beta, \eta \xleftarrow{\$} \mathbb{Z}_p$ uniformly at random.

Key Queries. On j^{th} secret key query x ($j \le q_1$), outputs type-3 secret key $\mathbf{K} \leftarrow \mathsf{SFKeyGen}(x, msk, -, \widehat{mpk}_{base}, 3, \beta_j)$ after choosing $\beta_j \xleftarrow{\$} \mathbb{Z}_p$.

Dec Queries. On j^{th} decryption query $(x, \overline{\mathbf{C}})$ where $\overline{\mathbf{C}}$ is a ciphertext on data-index y, if the signature σ is not verified or if $R(x, y) \neq 1$, aborts. Otherwise $\mathcal{B}$ computes altKey $\hat{\mathbf{K}}$ and returns $\mathsf{AltDec}(\overline{\mathbf{C}}, \hat{\mathbf{K}})$ to $\mathcal{A}$. We now describe the altKey generation procedure.

- If $j > i$, it is normal altKey. As $\mathcal{B}$ knows msk, it computes the altKey $\hat{\mathbf{K}} \leftarrow \mathsf{AltKeyGen}(\overline{\mathbf{C}}, x, msk)$.
- If $j < i$, it is type-3 semi-functional altKey. $\mathcal{B}$ computes type-3 altKey $\hat{\mathbf{K}} \leftarrow \mathsf{SFAltKeyGen}(\overline{\mathbf{C}}, x, msk, -, \widehat{mpk}_{base}, 3, \eta)$.
- If $j = i$, it runs $(\mathbf{k} = (k_1, \ldots, k_{m_1}); m_2) \leftarrow \mathsf{EncK}(x, N)$ and $\mathbf{E} \leftarrow \mathsf{Pair}(x, y, N)$. Chooses $\mathbf{r}_1, \ldots, \mathbf{r}_{m_2} \xleftarrow{\$} \mathbb{Z}_p^d$ and defines $\mathbf{R} = \left(\begin{pmatrix}\mathbf{r}_1\\ 0\end{pmatrix}, \ldots, \begin{pmatrix}\mathbf{r}_{m_2}\\ 0\end{pmatrix}\right)$. It generates normal key $\mathbf{K} = (\mathbf{K}_1, \ldots, \mathbf{K}_{m_1})$ where for each $\iota \in [m_1]$, $\mathbf{K}_\iota = g_2^{k_\iota(\boldsymbol{\alpha}, \mathbf{R}, \mathbb{H})} = g_2^{\left(b_\iota\boldsymbol{\alpha} + \sum\limits_{j\in[m_2]} b_{\iota j}\mathbf{Z}\begin{pmatrix}\mathbf{r}_j\\ 0\end{pmatrix} + \sum\limits_{\substack{j\in[m_2]\\ k\in[n]}} b_{\iota jk}\mathbf{H}_k^\top\mathbf{Z}\begin{pmatrix}\mathbf{r}_j\\ 0\end{pmatrix}\right)}$. It then computes $(\tilde{K}_1, \ldots, \tilde{K}_{w_1})$ where $\tilde{K}_{\tilde{\iota}} = \prod\limits_{\iota\in[m_1]} (\mathbf{K}_\iota)^{\mathbf{E}_{\iota\tilde{\iota}}}$ for each $\tilde{\iota} \in [w_1]$.

  Given $\overline{\mathbf{C}} = (\overline{\mathbf{C}}_0, \mathbf{C}, vk, \sigma)$ it computes $\xi = \mathcal{H}(\mathbf{C}, vk)$. To compute the altKey, it implicitly sets $\tilde{\mathbf{Z}}^{-1}\begin{pmatrix}\boldsymbol{y}\\ \hat{y}\end{pmatrix} = \begin{pmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{pmatrix}$. Therefore $g_2^{\mathbf{Z}\begin{pmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{pmatrix}} = g_2^{\tilde{\mathbf{B}}^{-\top}\mathbf{T}\begin{pmatrix}\boldsymbol{y}\\ \hat{y}\end{pmatrix}}$. Then the modified key is $\hat{\mathbf{K}} = (K_0, \Phi \cdot \tilde{K}_1, \tilde{K}_2, \ldots, \tilde{K}_{w_1})$ where $K_0 = g_2^{\mathbf{Z}\begin{pmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{pmatrix}}$, $\Phi = g_2^{(\xi\mathbf{H}_{n+1}^\top + \mathbf{H}_{n+2}^\top)\mathbf{Z}\begin{pmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{pmatrix}}$ and therefore is efficiently computable. It is evident from the description that if $\hat{y} = 0$, the key is a normal altKey whereas if $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$, the key is type-1 altKey.

Challenge. On receiving the challenge (y^*, M_0, M_1), $\mathcal{B}$ picks $\mathfrak{b} \xleftarrow{\$} \{0, 1\}$. It runs $(\mathbf{c} = (c_1, \ldots, c_{w_1}); w_2) \leftarrow \mathsf{EncC}(y^*, N)$. For each $j \in [0, w_2]$ it chooses $\begin{pmatrix}\mathbf{s}'_j\\ \hat{s}'_j\end{pmatrix} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)}$ and implicitly sets $\begin{pmatrix}\mathbf{s}_j\\ \hat{s}_j\end{pmatrix} = \mathbf{B}^{-1}\begin{pmatrix}\mathbf{s}'_j\\ \hat{s}'_j\end{pmatrix}$. Then $\mathcal{B}$ computes $\mathbf{C}^*$ as it knows $\boldsymbol{\alpha}$, $\mathbf{H}_1, \ldots, \mathbf{H}_{n+2}$. Runs $(vk^*, sk^*) \leftarrow \mathsf{OTS.Gen}(1^\lambda)$. Then it evaluates $\xi^* = \mathcal{H}(\mathbf{C}^*, vk^*)$ to compute $\overline{\mathbf{C}}^* = (\overline{\mathbf{C}}_0^*, \mathbf{C}^*, vk^*, \sigma^*)$ where $\overline{\mathbf{C}}_0^* = g_1^{(\xi^*\mathbf{H}_{n+1} + \mathbf{H}_{n+2})\begin{pmatrix}\mathbf{s}'_0\\ \hat{s}'_0\end{pmatrix}}$ and $\sigma^* \leftarrow \mathsf{OTS.Sign}(sk^*, \overline{\mathbf{C}}_0)$. It outputs $\overline{\mathbf{C}}^*$.

Key Queries. On secret key query x, outputs type-3 secret key $\mathbf{K} \leftarrow \mathsf{SFKeyGen}(x, msk, -, \widehat{mpk}_{base}, 3, \beta)$.

Dec Queries. Same as Phase-I decryption query.

Guess. $\mathcal{A}$ halts with output $\mathfrak{b}'$. $\mathcal{B}$ outputs 1 if $\mathfrak{b}' = \mathfrak{b}$ and 0 otherwise.

5.4 Type-1 to Type-2 altKey

Lemma 2 ($\mathsf{Game}_{3,i,1}$ **to** $\mathsf{Game}_{3,i,2}$)**.** *For $i = 1, \ldots, q_{_D}$, we have* $|\mathsf{Adv}^{3,i,1}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{3,i,2}_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\mathrm{CRH}}_{\mathcal{B}}(\lambda) + \mathsf{Adv}^{\mathrm{sUf\text{-}CMA}}_{\mathcal{B}',\mathsf{OTS}}(\lambda)$.

To prove the indistinguishability of the two games, we use the modified $\mathsf{SFSetup}$ namely $\mathsf{SFSetup}'$ (See Appendix B.3) that was used to prove indistinguishability of Lemma 7 and Lemma 10. Intuitively, to argue the indistinguishability, we introduce new randomness using $\mathsf{SFSetup}'$. Note that this newly introduced randomness does not affect the public key mpk. Then we show that introduction of such new randomness allows us to argue the indistinguishability. Recall that the challenge ciphertext is semi-functional and is denoted by $\overline{\mathbf{C}}^*$, the secret keys $\mathbf{K}$ are all type-3 keys and the altKey resulted from i^{th} decryption query is denoted by $\hat{\mathbf{K}}$.

Here we prove that joint distribution of $\{\mathbf{K}, \overline{\mathbf{C}}^*, \hat{\mathbf{K}}\}$ if $\hat{\mathbf{K}}$ is type-1 altKey is identical to joint distribution of $\{\mathbf{K}, \overline{\mathbf{C}}^*, \hat{\mathbf{K}}\}$ if $\hat{\mathbf{K}}$ is type-2 altKey. Note that $\overline{\mathbf{C}}^* = (\overline{\mathbf{C}}^*_0, \mathbf{C}^*, vk^*, \sigma^*)$ such that $\overline{\mathbf{C}}^*_0 = g_1^{(\xi^*\mathbf{H}_{n+1}+\mathbf{H}_{n+2})\mathbf{B}\begin{pmatrix}\mathbf{s}_0\\ \hat{s}_0\end{pmatrix}+\mathbf{B}\begin{pmatrix}\mathbf{0}\\ (\xi^*\hat{h}_{n+1}+\hat{h}_{n+2})\hat{s}_0\end{pmatrix}}$ where $\xi^* = \mathcal{H}(\mathbf{C}^*, vk^*)$ and $\sigma^* \leftarrow \mathsf{OTS.Sign}(sk^*, \overline{\mathbf{C}}^*_0)$ for $(vk^*, sk^*) \leftarrow \mathsf{OTS.Gen}(1^\lambda)$. Now we prove our claim that, the joint distributions of $\{\mathbf{K}, \overline{\mathbf{C}}^*, \hat{\mathbf{K}}\}$ behaves identically for both type-1 and type-2 altKey $\hat{\mathbf{K}}$.

Claim. The joint distribution of $\{\mathbf{K}, \overline{\mathbf{C}}^*, \hat{\mathbf{K}}\}$ if $\hat{\mathbf{K}}$ is type-1 altKey is identical to joint distribution of $\{\mathbf{K}, \overline{\mathbf{C}}^*, \hat{\mathbf{K}}\}$ if $\hat{\mathbf{K}}$ is type-2 altKey.

Proof. Note that $\mathbf{K}$ is type-3 key in both the distributions and can be computed by the simulator as it knows msk and $\widehat{mpk}_{\mathrm{base}}$. Due to linearity of pair encoding, the challenge ciphertext $\overline{\mathbf{C}}^*$ and the altKey $\hat{\mathbf{K}}$ can be expressed as product of normal component and semi-functional component. Since the simulator knows msk and can compute the normal components, it suffices to show that the joint distributions are identical if the semi-functional components of $\overline{\mathbf{C}}^*$ and $\hat{\mathbf{K}}$ are jointly identically distributed.

Notice that due to the introduction of $\hat{\mathbf{h}}$ (See Appendix B.3), the semi-functional ciphertext component $\overline{\mathbf{C}}^{*\prime}_0$ and the term Φ' used in altKey, is affected. To prove our claim, it suffices to argue that the following two distributions $(\overline{\mathbf{C}}^{*\prime}_0, \Phi')$ are identically distributed:

$$\left\{ g_1^{(\xi^*\mathbf{H}_{n+1}+\mathbf{H}_{n+2})\mathbf{B}\begin{pmatrix}\mathbf{0}\\ \hat{s}_0\end{pmatrix}+\mathbf{B}\begin{pmatrix}\mathbf{0}\\ (\xi^*\hat{h}_{n+1}+\hat{h}_{n+2})\hat{s}_0\end{pmatrix}},\ g_2^{\mathbf{Z}\begin{pmatrix}\mathbf{0}\\ (\xi\hat{h}_{n+1}+\hat{h}_{n+2})\hat{\mathfrak{r}}\end{pmatrix}+(\xi\mathbf{H}^\top_{n+1}+\mathbf{H}^\top_{n+2})\mathbf{Z}\begin{pmatrix}\mathbf{0}\\ \hat{\mathfrak{r}}\end{pmatrix}} \right\}$$

$$\left\{ g_1^{(\xi^*\mathbf{H}_{n+1}+\mathbf{H}_{n+2})\mathbf{B}\begin{pmatrix}\mathbf{0}\\ \hat{s}_0\end{pmatrix}+\mathbf{B}\begin{pmatrix}\mathbf{0}\\ (\xi^*\hat{h}_{n+1}+\hat{h}_{n+2})\hat{s}_0\end{pmatrix}},\ g_2^{\mathbf{Z}\begin{pmatrix}\mathbf{0}\\ u\eta\end{pmatrix}+\mathbf{Z}\begin{pmatrix}\mathbf{0}\\ (\xi\hat{h}_{n+1}+\hat{h}_{n+2})\hat{\mathfrak{r}}\end{pmatrix}+(\xi\mathbf{H}^\top_{n+1}+\mathbf{H}^\top_{n+2})\mathbf{Z}\begin{pmatrix}\mathbf{0}\\ \hat{\mathfrak{r}}\end{pmatrix}} \right\}$$

By natural restriction $\overline{\mathbf{C}}^* \neq \overline{\mathbf{C}}$ where $\overline{\mathbf{C}}^*$ is challenge ciphertext and $\overline{\mathbf{C}}$ is ciphertext on which decryption query is made. Therefore $(\overline{\mathbf{C}}^*_0, \mathbf{C}^*, vk^*, \sigma^*) \neq (\overline{\mathbf{C}}_0, \mathbf{C}, vk, \sigma)$.

Then any of the following two cases can happen,

1. If $(\mathbf{C}^*, vk^*) = (\mathbf{C}, vk)$, then we have found a forgery of the OTS namely $(\overline{\mathbf{C}}_0, \sigma) \neq (\overline{\mathbf{C}}^*_0, \sigma^*)$.
2. If $(\mathbf{C}^*, vk^*) \neq (\mathbf{C}, vk)$, then $\xi^* = \mathcal{H}(\mathbf{C}^*, vk^*)$ and $\xi = \mathcal{H}(\mathbf{C}, vk)$ are unequal due to collision resistance of $\mathcal{H}$. Therefore $\xi^*\hat{h}_{n+1} + \hat{h}_{n+2}$ and $\xi\hat{h}_{n+1} + \hat{h}_{n+2}$ are pairwise independent as $\hat{h}_{n+1}$ and $\hat{h}_{n+2}$ are chosen uniformly at random. It implies that the semi-functional components of the ciphertext and altKey in $\mathrm{Game}_{3,i,1}$ and $\mathrm{Game}_{3,i,2}$ are identically distributed.

5.5 Type-2 to Type-3 altKey

Lemma 3 ($\mathsf{Game}_{3,i,2}$ **to** $\mathsf{Game}_{3,i,3}$)**.** *For $i = 1, \ldots, q_{_D}$, for any efficient adversary $\mathcal{A}$ that makes at most q key queries and at most $q_{_D}$ decryption queries, there exists a PPT algorithm $\mathcal{B}$ such that* $|\mathsf{Adv}^{3,i-1,3}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{3,i,1}_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\mathcal{D}_d\text{-MatDH}}_{\mathcal{B}}(\lambda)$.

Proof. The algorithm $\mathcal{B}$ gets input $(\mathbf{G}, g_2^{\mathbf{T}}, g_2^{\mathbf{T}\begin{pmatrix}\boldsymbol{y}\\ \hat{y}\end{pmatrix}})$ as $\mathcal{D}_d$-MatDH problem instance where $\hat{y} = 0$ or $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$, $\boldsymbol{y} \xleftarrow{\$} \mathbb{Z}_p^d$.

Setup. $\mathcal{B}$ chooses $\tilde{\mathbf{B}} \xleftarrow{\$} \mathbb{GL}_{p,d+1}, \mathbf{J} \xleftarrow{\$} \mathbb{GL}_{p,d}$ and sets $\mathbf{B} = \tilde{\mathbf{B}}\begin{pmatrix}\mathbf{I}_d & \mathbf{M}^{-\top}\mathbf{c}^\top\\ \mathbf{0} & -1\end{pmatrix}$ and $\mathbf{D} = \begin{pmatrix}\mathbf{MJ} & \mathbf{0}\\ \mathbf{0} & 1\end{pmatrix}$ for $\mathbf{T} = \begin{pmatrix}\mathbf{M} & \mathbf{0}\\ \mathbf{c} & 1\end{pmatrix}$ due to $\mathcal{D}_d$-MatDH assumption. Then $\mathbf{Z} = \mathbf{B}^{-\top}\mathbf{D} = \tilde{\mathbf{B}}^{-\top}\begin{pmatrix}\mathbf{I}_d & \mathbf{0}\\ \mathbf{cM}^{-1} & -1\end{pmatrix}\begin{pmatrix}\mathbf{MJ} & \mathbf{0}\\ \mathbf{0} & 1\end{pmatrix} = \tilde{\mathbf{B}}^{-\top}\mathbf{T}\begin{pmatrix}\mathbf{J} & \mathbf{0}\\ \mathbf{0} & -1\end{pmatrix}$. Then defines $\tilde{\mathbf{Z}} = \begin{pmatrix}\mathbf{J} & \mathbf{0}\\ \mathbf{0} & -1\end{pmatrix}$ so that $\mathbf{Z} = \tilde{\mathbf{B}}^{-\top}\mathbf{T}\tilde{\mathbf{Z}}$. $\mathcal{B}$ therefore can compute the public parameters as $g_1^{\mathbf{B}\begin{pmatrix}\mathbf{I}_d\\ 0\end{pmatrix}} = g_1^{\tilde{\mathbf{B}}\begin{pmatrix}\mathbf{I}_d\\ 0\end{pmatrix}}$ and $g_2^{\mathbf{Z}} = g_2^{\tilde{\mathbf{B}}^{-\top}\mathbf{T}\tilde{\mathbf{Z}}}$. $\mathcal{B}$ then chooses $\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)}$ and $\mathbb{H} = (\mathbf{H}_1, \ldots, \mathbf{H}_{n+2}) \xleftarrow{\$} \left(\mathbb{Z}_p^{(d+1)\times(d+1)}\right)^{(n+2)}$ and publishes public key mpk. $\mathcal{B}$ cannot compute $\widehat{mpk}_b$ but can compute $\widehat{mpk}_z$ as it can compute $\widehat{mpk}_{base}$. It chooses $\beta, \eta \xleftarrow{\$} \mathbb{Z}_p$ uniformly at random.

Key Queries. On j^{th} secret key query x, outputs type-3 secret key $\mathbf{K} \leftarrow \mathsf{SFKeyGen}(x, msk, -, \widehat{mpk}_{base}, 3, \beta_j)$ after choosing $\beta_j \xleftarrow{\$} \mathbb{Z}_p$.

Dec Queries. On j^{th} decryption query $(x, \overline{\mathbf{C}})$ where $\overline{\mathbf{C}}$ is a ciphertext on data-index y, if the signature σ is not verified or if $R(x, y) \neq 1$, aborts. Otherwise $\mathcal{B}$ computes altKey $\hat{\mathbf{K}}$ and returns $\mathsf{AltDec}(\overline{\mathbf{C}}, \hat{\mathbf{K}})$ to $\mathcal{A}$. We now describe the altKey generation procedure.

- If $j > i$, it is normal altKey. As $\mathcal{B}$ knows msk, it computes the altKey $\hat{\mathbf{K}} \leftarrow \mathsf{AltKeyGen}(\overline{\mathbf{C}}, x, msk)$.
- If $j < i$, it is type-3 semi-functional altKey. $\mathcal{B}$ computes type-3 altKey $\hat{\mathbf{K}} \leftarrow \mathsf{SFAltKeyGen}(\overline{\mathbf{C}}, x, msk, -, \widehat{mpk}_{base}, 3, \eta)$.
- If $j = i$, it runs $(\mathbf{k} = (k_1, \ldots, k_{m_1}); m_2) \leftarrow \mathsf{EncK}(x, N)$ and $\mathbf{E} \leftarrow \mathsf{Pair}(x, y, N)$. Chooses $\mathbf{r}_1, \ldots, \mathbf{r}_{m_2} \xleftarrow{\$} \mathbb{Z}_p^d$ and defines $\mathbf{R} = \left(\begin{pmatrix}\mathbf{r}_1\\ 0\end{pmatrix}, \ldots, \begin{pmatrix}\mathbf{r}_{m_2}\\ 0\end{pmatrix}\right)$. It generates normal key $\mathbf{K} = (\mathbf{K}_1, \ldots, \mathbf{K}_{m_1})$ where for each $\iota \in [m_1]$, $\mathbf{K}_\iota = g_2^{k_\iota(\boldsymbol{\alpha}, \mathbf{R}, \mathbb{H})} = g_2^{\left(b_\iota\boldsymbol{\alpha} + \sum\limits_{j\in[m_2]} b_{\iota j}\mathbf{Z}\begin{pmatrix}\mathbf{r}_j\\ 0\end{pmatrix} + \sum\limits_{\substack{j\in[m_2]\\ k\in[n]}} b_{\iota jk}\mathbf{H}_k^\top\mathbf{Z}\begin{pmatrix}\mathbf{r}_j\\ 0\end{pmatrix}\right)}$. It then computes $(\tilde{K}_1, \ldots, \tilde{K}_{w_1})$ where $\tilde{K}_{\tilde{\iota}} = \prod\limits_{\iota\in[m_1]} (\mathbf{K}_\iota)^{\mathbf{E}_{\iota\tilde{\iota}}}$ for each $\tilde{\iota} \in [w_1]$.

  Given $\overline{\mathbf{C}} = (\overline{\mathbf{C}}_0, \mathbf{C}, vk, \sigma)$ it computes $\xi = \mathcal{H}(\mathbf{C}, vk)$. To compute the altKey, it implicitly sets $\tilde{\mathbf{Z}}^{-1}\begin{pmatrix}\boldsymbol{y}\\ \hat{y}\end{pmatrix} = \begin{pmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{pmatrix}$. Therefore $g_2^{\mathbf{Z}\begin{pmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{pmatrix}} = g_2^{\tilde{\mathbf{B}}^{-\top}\mathbf{T}\begin{pmatrix}\boldsymbol{y}\\ \hat{y}\end{pmatrix}}$. Then the modified key is $\hat{\mathbf{K}} = (K_0, \Phi \cdot \tilde{K}_1, \tilde{K}_2, \ldots, \tilde{K}_{w_1})$ where $K_0 = g_2^{\mathbf{Z}\begin{pmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{pmatrix}}$, $\Phi = g_2^{\mathbf{Z}\begin{pmatrix}\mathbf{0}\\ \eta u\end{pmatrix} + (\xi\mathbf{H}_{n+1}^\top + \mathbf{H}_{n+2}^\top)\mathbf{Z}\begin{pmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{pmatrix}}$ such that $u = \sum\limits_{\iota\in[m_1]} b_\iota\mathbf{E}_{\iota 1}$ and therefore is efficiently computable.

  It is evident from the description that if $\hat{y} = 0$, the key is a type-3 altKey whereas if $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$, the key is type-2 altKey.

Challenge. On receiving challenge (y^*, M_0, M_1), $\mathcal{B}$ picks $\mathfrak{b} \xleftarrow{\$} \{0, 1\}$. It runs $(\mathbf{c} = (c_1, \ldots, c_{w_1}); w_2) \leftarrow \mathsf{EncC}(y, N)$. For each $j \in [0, w_2]$ it chooses $\begin{pmatrix}\mathbf{s}'_j\\ \hat{s}'_j\end{pmatrix} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)}$ and implicitly sets $\begin{pmatrix}\mathbf{s}_j\\ \hat{s}_j\end{pmatrix} = \mathbf{B}^{-1}\begin{pmatrix}\mathbf{s}'_j\\ \hat{s}'_j\end{pmatrix}$.

Then $\mathcal{B}$ computes $\mathbf{C}^*$ as it knows $\boldsymbol{\alpha}$, $\mathbf{H}_1, \ldots, \mathbf{H}_{n+2}$. Runs $(vk^*, sk^*) \leftarrow \mathsf{OTS.Gen}(1^\lambda)$. Then it evaluates $\xi^* = \mathcal{H}(\mathbf{C}^*, vk^*)$ to compute $\overline{\mathbf{C}}^* = (\overline{\mathbf{C}}_0^*, \mathbf{C}^*, vk^*, \sigma^*)$ where $\overline{\mathbf{C}}_0^* = g_1^{(\xi^*\mathbf{H}_{n+1} + \mathbf{H}_{n+2})\begin{pmatrix}\mathbf{s}'_0\\ \hat{s}'_0\end{pmatrix}}$ and $\sigma^* \leftarrow \mathsf{OTS.Sign}(sk^*, \overline{\mathbf{C}}_0)$. It outputs $\overline{\mathbf{C}}^*$.

Key Queries. On secret key query x, outputs type-3 secret key $\mathbf{K} \leftarrow \mathsf{SFKeyGen}(x, msk, -, \widehat{mpk}_{base}, 3, \beta)$.

Dec Queries. Same as Phase-I decryption query.

Guess. $\mathcal{A}$ halts with output $\mathfrak{b}'$. $\mathcal{B}$ outputs 1 if $\mathfrak{b}' = \mathfrak{b}$ and 0 otherwise.

5.6 Final Game

Lemma 4 ($\mathsf{Game}_{3,q_D,3}$ **to** Game_4)**.** *For any adversary* $\mathcal{A}$, $|\mathsf{Adv}^{3,q_D,3}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{4}_{\mathcal{A}}(\lambda)| = 0$.

Proof. As $\mathbf{Z} \in \mathbb{GL}_{p,d+1}$, one can express $\boldsymbol{\alpha}$ as a linear combination of column vectors of $\mathbf{Z}$ i.e. $\boldsymbol{\alpha} = \mathbf{Z}\begin{pmatrix}\boldsymbol{\delta}\\ \hat{\delta}\end{pmatrix}$ for $\boldsymbol{\delta} \in \mathbb{Z}_p^d$ and $\hat{\delta} \in \mathbb{Z}_p$. In all the secret keys, $\hat{\delta}$ is hidden by uniformly random β_i (in case of pre-challenge secret key queries) and by uniformly random β (in case of post-challenge key queries). Note that in case of altKeys, the presence of $\boldsymbol{\alpha}$ is limited only to $\hat{\mathbf{K}}_1$ due to regular decryption property of pair encoding (precisely Property $\mathcal{P}5$) in the form of $u\boldsymbol{\alpha}$ where $u = \sum_{\iota \in [m_1]} b_\iota \mathbf{E}_{\iota 1}$. The term, $u\mathbf{Z}\begin{pmatrix}\mathbf{0}\\ \eta\end{pmatrix}$, appears in the exponent of Φ of type-3 altKeys. Therefore in all altKeys, $\hat{\delta}$ of $\boldsymbol{\alpha}$ will be is hidden by uniformly random η.

Therefore we can replace $\hat{\delta}$ by $\hat{\delta} + t$ for $t \xleftarrow{\$} \mathbb{Z}_p$. Notice that such a change will affect the ciphertext in only one component namely $\mathbf{C}^*_{w_1+1}$. The resultant $\mathbf{C}^*_{w_1+1}$ will be $M_{\mathfrak{b}} \cdot e(g_1,g_2)^{\boldsymbol{\alpha}^\top \mathbf{B}\begin{pmatrix}\mathbf{s}_0\\ \hat{s}_0\end{pmatrix}} = M_{\mathfrak{b}} \cdot e(g_1,g_2)^{(\boldsymbol{\delta}^\top\ \hat{\delta}+t)\mathbf{Z}^\top \mathbf{B}\begin{pmatrix}\mathbf{s}_0\\ \hat{s}_0\end{pmatrix}} = M_{\mathfrak{b}} \cdot e(g_1,g_2)^{\boldsymbol{\alpha}^\top \mathbf{B}\begin{pmatrix}\mathbf{s}_0\\ \hat{s}_0\end{pmatrix}} \cdot e(g_1,g_2)^{(\mathbf{0}\ t)\mathbf{Z}^\top \mathbf{B}\begin{pmatrix}\mathbf{s}_0\\ \hat{s}_0\end{pmatrix}} = M_{\mathfrak{b}} \cdot e(g_1,g_2)^{\boldsymbol{\alpha}^\top \mathbf{B}\begin{pmatrix}\mathbf{s}_0\\ \hat{s}_0\end{pmatrix}} \cdot e(g_1,g_2)^{t\hat{s}_0}$. Therefore $\mathbf{C}^*_{w_1+1}$ encrypts $M_{\mathfrak{b}} \cdot e(g_1,g_2)^{t\hat{s}_0}$ that is an uniformly random element of $\mathbb{G}_T$ as $t \xleftarrow{\$} \mathbb{Z}_p$.

6 Conclusion

We generically converted the adaptive CPA-secure predicate encryption of [Att16] to adaptive CCA-secure predicate encryption. The ciphertext of our adaptive CCA-secure predicate encryption contains *exactly three* additional components (a $\mathbb{G}_1^{(d+1)}$ element, an OTS verification key and a signature) than in case of adaptive CPA-secure predicate encryption of [Att16]. To verify a ciphertext, one needs only $(d+1)$ additional pairing evaluations in our construction apart from the verification of the signature in the ciphertext. This is a significant improvement over the previous generic conversion mechanisms which needed almost double of $m_1 \times w_1 \times (d+1) \times (m_2+1) \times d$ many pairing evaluations. A possible future work might be instantiation of our generic CPA-to-CCA conversion on the predicate encryption resulted from integration of *dual system groups* with *pair encoding schemes*.

References

[ABS16] Miguel Ambrona, Gilles Barthe, and Benedikt Schmidt. Generic transformations of predicate encodings: Constructions and applications. Cryptology ePrint Archive, Report 2016/1105, 2016. http://eprint.iacr.org/2016/1105.

[AC16] Shashank Agrawal and Melissa Chase. *A Study of Pair Encodings: Predicate Encryption in Prime Order Groups*, pages 259–288. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[AC17] Shashank Agrawal and Melissa Chase. Simplifying design and analysis of complex predicate encryption schemes. Technical report, IACR Cryptology ePrint Archive, 2017.

[Att14] Nuttapong Attrapadung. *Dual System Encryption via Doubly Selective Security: Framework, Fully Secure Functional Encryption for Regular Languages, and More*, pages 557–577. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[Att16] Nuttapong Attrapadung. *Dual System Encryption Framework in Prime-Order Groups via Computational Pair Encodings*, pages 591–623. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *Advances in Cryptology CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer Berlin Heidelberg, 2001.

[BH08] Dan Boneh and Michael Hamburg. Generalized identity based and broadcast encryption schemes. In Josef Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 455–470. Springer Berlin Heidelberg, 2008.

[BL16] Johannes Blömer and Gennadij Liske. *Construction of Fully CCA-Secure Predicate Encryptions from Pair Encoding Schemes*, pages 431–447. Springer International Publishing, Cham, 2016.

[BMW05] Xavier Boyen, Qixiang Mei, and Brent Waters. Direct chosen ciphertext security from identity-based techniques. In *Proceedings of the 12th ACM Conference on Computer and Communications Security*, CCS '05, pages 320–329, New York, NY, USA, 2005. ACM.

[BSW07] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy (SP '07)*, pages 321–334, May 2007.

[CGW15] Jie Chen, Romain Gay, and Hoeteck Wee. *Improved Dual System ABE in Prime-Order Groups via Predicate Encodings*, pages 595–624. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[CW14] Jie Chen and Hoeteck Wee. *Dual System Groups and its Applications-Compact HIBE and More. IACR Cryptology ePrint Archive*, 2014:265, 2014.

[EHK+17] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for diffie–hellman assumptions. *Journal of Cryptology*, 30(1):242–288, 2017.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 89–98, New York, NY, USA, 2006. ACM.

[Ham11] Mike Hamburg. Spatial encryption. Technical report, IACR Cryptology ePrint Archive, 2011.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel Smart, editor, *Advances in Cryptology EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162. Springer Berlin Heidelberg, 2008.

[LW10] Allison Lewko and Brent Waters. *New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts*, pages 455–479. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[NP15a] Mridul Nandi and Tapas Pandit. Generic conversions from CPA to CCA secure functional encryption. *IACR Cryptology ePrint Archive*, 2015:457, 2015.

[NP15b] Mridul Nandi and Tapas Pandit. On the power of pair encodings: Frameworks for predicate cryptographic primitives. *IACR Cryptology ePrint Archive*, 2015:955, 2015.

[NP17] Mridul Nandi and Tapas Pandit. Verifiability-based conversion from cpa to cca-secure predicate encryption. *Applicable Algebra in Engineering, Communication and Computing*, Jun 2017.

[OT09] Tatsuaki Okamoto and Katsuyuki Takashima. *Hierarchical Predicate Encryption for Inner-Products*, pages 214–231. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In GeorgeRobert Blakley and David Chaum, editors, *Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer Berlin Heidelberg, 1985.

[SSW09] Emily Shen, Elaine Shi, and Brent Waters. *Predicate Privacy in Encryption Systems*, pages 457–473. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[Wat09] Brent Waters. *Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions*, pages 619–636. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[Wat11] Brent Waters. *Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization*, pages 53–70. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[Wee14] Hoeteck Wee. *Dual System Encryption via Predicate Encodings*, pages 616–637. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[YAHK11] Shota Yamada, Nuttapong Attrapadung, Goichiro Hanaoka, and Noboru Kunihiro. *Generic Constructions for Chosen-Ciphertext Secure Attribute Based Encryption*, pages 71–89. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[YAS+12] Shota Yamada, Nuttapong Attrapadung, Bagus Santoso, Jacob C. N. Schuldt, Goichiro Hanaoka, and Noboru Kunihiro. *Verifiable Predicate Encryption and Applications to CCA Security and Anonymous Predicate Authentication*, pages 243–261. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

A Conventional approach to achieve CCA-secure Predicate encryption

The predicate encryption schemes presented in [Att16] are CPA-secure. We already have pointed out one can convert these schemes to achieve CCA-security by incorporating the generic conversion framework of [YAHK11,YAS+12,NP17]. Here we show that pair encoding based predicate encryption schemes instantiated in prime-order groups [Att16] fulfills the notion of *verifiability* [YAHK11]. Intuitively, a predicate encryption scheme has *verifiability* if there exists a procedure that confirms if a ciphertext decrypts to same message under two different keys. Here we define the algorithm Verify as following for $\mathbf{C}$ being a ciphertext corresponding to data-index y and two different key-indices x and $\tilde{x}$. Let $\mathbf{k} = (k_1, \ldots, k_{m_1})$ and $\tilde{\mathbf{k}} = (\tilde{k}_1, \ldots, \tilde{k}_{\tilde{m}_1})$ be the output of EncK on input x and $\tilde{x}$ respectively. We also denote corresponding secret keys by $\mathbf{K}$ and $\tilde{\mathbf{K}}$ respectively. Let $\mathbf{E} = \mathsf{Pair}(x, y, N)$ and $\tilde{\mathbf{E}} = \mathsf{Pair}(\tilde{x}, y, N)$.

$$\mathsf{Verify}(pk, \mathbf{C}, x, \tilde{x}) = \begin{cases} \perp & \text{if } R(x,y) = 0 \text{ or } R(\tilde{x}, y) = 0 \\ 1 & \text{if } \mathsf{Event} \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathsf{Event} = \begin{cases} \prod\limits_{\substack{\iota\in[m_1]\\ \tilde{\iota}\in[w_1]}} e\Big(\mathbf{C}_{\tilde{\iota}}, g_2^{k_\iota(\mathbf{0},\mathbb{I}_{tj},\mathbb{H})}\Big)^{\mathbf{E}_{\iota\tilde{\iota}}} = 1 \text{ for all } t\in[d],\ j\in[m_2] & (2) \\ \prod\limits_{\substack{\iota\in[\tilde{m}_1]\\ \tilde{\iota}\in[w_1]}} e\Big(\mathbf{C}_{\tilde{\iota}}, g_2^{\tilde{k}_\iota(\mathbf{0},\mathbb{I}_{tj},\mathbb{H})}\Big)^{\tilde{\mathbf{E}}_{\iota\tilde{\iota}}} = 1 \text{ for all } t\in[d],\ j\in[\tilde{m}_2] & (3) \\ \prod\limits_{\substack{\iota\in[m_1]\\ \tilde{\iota}\in[w_1]}} e\big(\mathbf{C}_{\tilde{\iota}}, g_2^{b_\iota \mathbf{1}_t}\big)^{\mathbf{E}_{\iota\tilde{\iota}}} = \prod\limits_{\substack{\iota\in[\tilde{m}_1]\\ \tilde{\iota}\in[w_1]}} e\Big(\mathbf{C}_{\tilde{\iota}}, g_2^{\tilde{b}_\iota \mathbf{1}_t}\Big)^{\tilde{\mathbf{E}}_{\iota\tilde{\iota}}} = e(\mathbf{C}_1, g_2^{\mathbf{1}_t}) \text{ for all } t\in[d]. & (4) \end{cases}$$

where $\mathbb{I}_{tj}$ is a sparse matrix whose $(t,j)^{th}$ entry alone is 1; $\mathbf{0}$ is a vector of length $(d+1)$ with all entries being zero and $\mathbf{1}_t$ is a sparse vector of length $(d+1)$ whose t^{th} entry alone is 1.

Completeness of Verifiability. Suppose the ciphertext $\mathbf{C}$ is correctly generated for data-index y. We need to show that for x, $\tilde{x}$ such that $R(x,y) = 1$ and $R(\tilde{x}, y) = 1$, $\mathsf{Verify}(pk, \mathbf{C}, x, \tilde{x}) = 1$. Here due to correctness of the predicate encryption of [Att16], all the equations $\big($namely Eq. $(2), (3), (4)\big)$ hold true. Note that the correctness of predicate encryption construction of [Att16] required only Property $\mathcal{P}1$ of regular decryption properties (Sec. 4.1) of underlying pair encoding. However, to satisfy Eq. (4), a well-formed ciphertext will also require Property $\mathcal{P}4$ of regular decryption properties of underlying pair encoding.

Soundness of Verifiability. Assume for $x, \tilde{x}$ and y, $R(x,y) = 1$ and $R(\tilde{x}, y) = 1$. If $\mathsf{Verify}(pk, \mathbf{C}, x, \tilde{x}) = 1$, we show that $\mathsf{Decrypt}(pk, \mathbf{C}, \mathbf{K})$ and $\mathsf{Decrypt}(pk, \mathbf{C}, \tilde{\mathbf{K}})$ outputs the same. Let $\Delta := \mathsf{Decrypt}(pk, \mathbf{C}, \mathbf{K})$.

By the definition of pair encoding,

$$k_\iota(\boldsymbol{\alpha}, \mathbb{O}, \mathbb{H}) = b_\iota \boldsymbol{\alpha} = \sum_{t\in[d+1]} \alpha_t (b_\iota \mathbf{1}_t), \tag{5}$$

$$k_\iota(\mathbf{0}, \mathbb{R}, \mathbb{H}) = \sum_{\substack{t\in[d]\\ j\in[m_2]}} r_{tj} k_\iota(\mathbf{0}, \mathbb{I}_{tj}, \mathbb{H}). \tag{6}$$

Then $\Delta = \mathsf{Decrypt}(pk, \mathbf{C}, \mathbf{K}) = \mathbf{C}_{w_1+1} / \prod\limits_{\substack{\iota\in[m_1]\\ \tilde{\iota}\in[w_1]}} e\left(\mathbf{C}_{\tilde{\iota}}, g_2^{k_\iota(\boldsymbol{\alpha},\mathbb{R},\mathbb{H})}\right)^{\mathbf{E}_{\iota\tilde{\iota}}}$.

We define, $\mathfrak{d} = \prod\limits_{\substack{\iota\in[m_1]\\ \tilde{\iota}\in[w_1]}} e\left(\mathbf{C}_{\tilde{\iota}}, g_2^{k_\iota(\boldsymbol{\alpha},\mathbb{R},\mathbb{H})}\right)^{\mathbf{E}_{\iota\tilde{\iota}}}$

$$= \prod_{\substack{\iota\in[m_1]\\ \tilde{\iota}\in[w_1]}} e\left(\mathbf{C}_{\tilde{\iota}}, g_2^{k_\iota(\boldsymbol{\alpha},\mathbb{O},\mathbb{H}) + k_\iota(\mathbf{0},\mathbb{R},\mathbb{H})}\right)^{\mathbf{E}_{\iota\tilde{\iota}}} \qquad \text{(by linearity)}$$

$$= \prod_{\substack{\iota\in[m_1]\\ \tilde{\iota}\in[w_1]}} e\left(\mathbf{C}_{\tilde{\iota}}, g_2^{k_\iota(\boldsymbol{\alpha},\mathbb{O},\mathbb{H})}\right)^{\mathbf{E}_{\iota\tilde{\iota}}} \cdot \prod_{\substack{\iota\in[m_1]\\ \tilde{\iota}\in[w_1]}} e\left(\mathbf{C}_{\tilde{\iota}}, g_2^{k_\iota(\mathbf{0},\mathbb{R},\mathbb{H})}\right)^{\mathbf{E}_{\iota\tilde{\iota}}}$$

$$= \mathfrak{A} \cdot \mathfrak{B}$$

where $\mathfrak{A} = \prod\limits_{\substack{\iota\in[m_1]\\ \tilde{\iota}\in[w_1]}} e\left(\mathbf{C}_{\tilde{\iota}}, g_2^{k_\iota(\boldsymbol{\alpha},\mathbb{O},\mathbb{H})}\right)^{\mathbf{E}_{\iota\tilde{\iota}}}$ and $\mathfrak{B} = \prod\limits_{\substack{\iota\in[m_1]\\ \tilde{\iota}\in[w_1]}} e\left(\mathbf{C}_{\tilde{\iota}}, g_2^{k_\iota(\mathbf{0},\mathbb{R},\mathbb{H})}\right)^{\mathbf{E}_{\iota\tilde{\iota}}}$.

$$\text{Now } \mathfrak{A} = \prod_{\substack{\iota\in[m_1]\\ \tilde{\iota}\in[w_1]}} e\left(\mathbf{C}_{\tilde{\iota}}, g_2^{k_\iota(\boldsymbol{\alpha},\mathbb{O},\mathbb{H})}\right)^{\mathbf{E}_{\iota\tilde{\iota}}}$$

$$= \prod_{\substack{\iota\in[m_1]\\ \tilde{\iota}\in[w_1]}} e\left(\mathbf{C}_{\tilde{\iota}}, g_2^{\sum\limits_{t\in[d+1]} \alpha_t(b_\iota \mathbf{1}_t)}\right)^{\mathbf{E}_{\iota\tilde{\iota}}} \qquad \text{(by Eq. (5))}$$

$$= \prod_{t\in[d+1]} \left(\prod_{\substack{\iota\in[m_1]\\ \tilde{\iota}\in[w_1]}} e\left(\mathbf{C}_{\tilde{\iota}}, g_2^{\alpha_t(b_\iota \mathbf{1}_t)}\right)^{\mathbf{E}_{\iota\tilde{\iota}}} \right)$$

$$= \prod_{t\in[d+1]} \left(\prod_{\substack{\iota\in[m_1]\\ \tilde{\iota}\in[w_1]}} e\left(\mathbf{C}_{\tilde{\iota}}, g_2^{b_\iota \mathbf{1}_t}\right)^{\mathbf{E}_{\iota\tilde{\iota}}} \right)^{\alpha_t}$$

$$= \prod_{t\in[d+1]} \left(e(\mathbf{C}_1, g_2^{\mathbf{1}_t})\right)^{\alpha_t} \qquad \text{(by Eq. (4))}$$

$$= e(\mathbf{C}_1, g_2^{\boldsymbol{\alpha}}).$$

$$\text{And } \mathfrak{B} = \prod_{\substack{\iota\in[m_1]\\ \tilde{\iota}\in[w_1]}} e\left(\mathbf{C}_{\tilde{\iota}}, g_2^{k_\iota(\mathbf{0},\mathbb{R},\mathbb{H})}\right)^{\mathbf{E}_{\iota\tilde{\iota}}}$$

$$= \prod_{\substack{\iota\in[m_1]\\ \tilde{\iota}\in[w_1]}} e\left(\mathbf{C}_{\tilde{\iota}}, g_2^{\sum\limits_{\substack{t\in[d]\\ j\in[m_2]}} r_{tj} k_\iota(\mathbf{0},\mathbb{I}_{tj},\mathbb{H})}\right)^{\mathbf{E}_{\iota\tilde{\iota}}} \qquad \text{(by Eq. (6))}$$

$$= \prod_{\substack{t\in[d]\\ j\in[m_2]}} \left(\prod_{\substack{\iota\in[m_1]\\ \tilde{\iota}\in[w_1]}} e\left(\mathbf{C}_{\tilde{\iota}}, g_2^{k_\iota(\mathbf{0},\mathbb{I}_{tj},\mathbb{H})}\right)^{\mathbf{E}_{\iota\tilde{\iota}}} \right)^{r_{tj}}$$

$$= \prod_{\substack{t\in[d]\\ j\in[m_2]}} (1)^{r_{tj}} = 1. \qquad \text{(by Eq. (2))}$$

As $\mathfrak{d} = \mathfrak{A} \cdot \mathfrak{B} = e(\mathbf{C}_1, g_2^{\boldsymbol{\alpha}})$, $\Delta = \mathbf{C}_{w_1+1}/\mathfrak{d} = \mathbf{C}_{w_1+1}/e(\mathbf{C}_1, g_2^{\boldsymbol{\alpha}})$.

Since x is arbitrary, similarly we have $\mathsf{Decrypt}(pk,\ \mathbf{C}, \tilde{\mathbf{K}}) = \mathbf{C}_{w_1+1}/e(\mathbf{C}_1, g_2^{\boldsymbol{\alpha}})$.

Hence we note that [Att16] schemes achieves *verifiability* and can be converted generically to achieve CCA-security [YAHK11,YAS+12,NP17]. We also note that $(m_1 \times w_1 \times (d+1) \times (m_2+1) \times d) + (\tilde{m}_1 \times w_1 \times (d+1) \times (\tilde{m}_2+1) \times d) + (d+1) \times d$ many additional pairing computations were needed to verify the well-formedness of the queried ciphertext.

Remark 5. This count actually is loose upper bound as the matrix $\mathbf{E}$ and $\tilde{\mathbf{E}}$ are usually sparse. The actual number of additional pairing to be evaluated is $\left(I \times (m_2+1) + \tilde{I} \times (\tilde{m}_2+1) + 1\right) \times d \times (d+1)$ where I and $\tilde{I}$ are the numbers of non-zero entries in $\mathbf{E}$ and $\tilde{\mathbf{E}}$ respectively. Note that this is still quite a large number as opposed to our achievement of $(d+1)$ (presented in Section 4.2) additional pairings only.

B Security Proof Details

B.1 Normal to Semi-functional Ciphertext

Lemma 5 (Game_0 **to** Game_1)**.** *For any efficient adversary $\mathcal{A}$ that makes at most q_1 pre-challenge key queries, at most q_2 post-challenge key queries and at most $q_{_D}$ decryption queries, there exists a PPT algorithm $\mathcal{B}$ such that $|\mathsf{Adv}^0_{\mathcal{A}}(\lambda) - \mathsf{Adv}^1_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\mathcal{D}_d\text{-MatDH}}_{\mathcal{B}}(\lambda)$.*

Proof. The algorithm $\mathcal{B}$ gets input $(\mathbf{G}, g_1^{\mathbf{T}}, g_1^{\mathbf{T}\begin{pmatrix}\boldsymbol{y}\\ \hat{y}\end{pmatrix}})$ as $\mathcal{D}_d$-MatDH problem instance where $\hat{y} = 0$ or $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$, $\boldsymbol{y} \xleftarrow{\$} \mathbb{Z}_p^d$.

Setup. $\mathcal{B}$ chooses $\tilde{\mathbf{B}} \xleftarrow{\$} \mathbb{GL}_{p,d+1}, \mathbf{J} \xleftarrow{\$} \mathbb{GL}_{p,d}$ and implicitly sets $\mathbf{B} = \tilde{\mathbf{B}}\mathbf{T}$ and $\mathbf{Z} = \tilde{\mathbf{B}}^{-\top}\begin{pmatrix}\mathbf{J} & -\mathbf{M}^{-\top}\mathbf{c}^\top\\ \mathbf{0} & 1\end{pmatrix}$ such that $\mathbf{D} = \mathbf{B}^\top\mathbf{Z} = \begin{pmatrix}\mathbf{T}^\top & \tilde{\mathbf{B}}^\top\end{pmatrix}\left(\tilde{\mathbf{B}}^{-\top}\begin{pmatrix}\mathbf{J} & -\mathbf{M}^{-\top}\mathbf{c}^\top\\ \mathbf{0} & 1\end{pmatrix}\right) = \begin{pmatrix}\mathbf{M}^\top & \mathbf{c}^\top\\ \mathbf{0} & 1\end{pmatrix}\begin{pmatrix}\mathbf{J} & -\mathbf{M}^{-\top}\mathbf{c}^\top\\ \mathbf{0} & 1\end{pmatrix} = \begin{pmatrix}\mathbf{M}^\top\mathbf{J} & \mathbf{0}\\ \mathbf{0} & 1\end{pmatrix}$. $\mathcal{B}$ can compute $g_1^{\mathbf{B}} = g_1^{\tilde{\mathbf{B}}\mathbf{T}}$ and $g_2^{\mathbf{Z}\begin{pmatrix}\mathbf{I}_d\\ 0\end{pmatrix}} = g_2^{\tilde{\mathbf{B}}^{-\top}\begin{pmatrix}\mathbf{J}\\ 0\end{pmatrix}}$. Therefore it can easily compute mpk, msk by choosing the parameters $\boldsymbol{\alpha}, \mathbf{H}_1, \ldots, \mathbf{H}_{n+2}$ itself.

Key Queries. On secret key query x, outputs secret key $\mathbf{K} \leftarrow \mathsf{KeyGen}(x, msk)$.

Dec Queries. On decryption query $(x, \overline{\mathbf{C}})$ where $\overline{\mathbf{C}}$ is a ciphertext on data-index y, if the signature σ is not verified or if $R(x, y) \neq 1$, aborts. Otherwise $\mathcal{B}$ computes normal altKey $\hat{\mathbf{K}} \leftarrow \mathsf{AltKeyGen}(\overline{\mathbf{C}}, x, msk)$ and returns $\mathsf{AltDec}(\overline{\mathbf{C}}, \hat{\mathbf{K}})$ to $\mathcal{A}$.

Challenge. On receiving the challenge (y^*, M_0, M_1), $\mathcal{B}$ picks $\mathfrak{b} \xleftarrow{\$} \{0,1\}$. Let $(\mathbf{c} = (c_1, \ldots, c_{w_1}); w_2) \leftarrow \mathsf{EncC}(y^*, N)$. It uses random self-reducibility of Matrix-DH assumption to obtain $(\mathbf{G}, g_1^{\mathbf{T}}, g_1^{\mathbf{T}\begin{pmatrix}\mathbf{Y}\\ \hat{\boldsymbol{y}}\end{pmatrix}})$. The decision problem is now to find if $\hat{\boldsymbol{y}} = \mathbf{0}$ or $\hat{\boldsymbol{y}} \xleftarrow{\$} \left(\mathbb{Z}_p\right)^{(w_2+1)}$ where $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$ and $\mathbf{Y} \xleftarrow{\$} \left(\mathbb{Z}_p^d\right)^{(w_2+1)}$. $\mathcal{B}$ implicitly sets $\begin{pmatrix}\mathbf{Y}\\ \hat{\boldsymbol{y}}\end{pmatrix} = \mathbf{S} + \hat{\mathbf{S}} = \begin{pmatrix}\mathbf{s}_0 & \cdots & \mathbf{s}_{w_2}\\ \hat{s}_0 & \cdots & \hat{s}_{w_2}\end{pmatrix}$. As $\mathcal{B}$ has $g_1^{\mathbf{T}\begin{pmatrix}\mathbf{Y}\\ \hat{\boldsymbol{y}}\end{pmatrix}}$, it can compute $g_1^{\mathbf{B}\begin{pmatrix}\mathbf{s}_j\\ \hat{s}_j\end{pmatrix}} = g_1^{\tilde{\mathbf{B}}\mathbf{T}\begin{pmatrix}\mathbf{s}_j\\ \hat{s}_j\end{pmatrix}} = g_1^{\tilde{\mathbf{B}}\mathbf{T}\begin{pmatrix}\boldsymbol{y}_j\\ \hat{y}_j\end{pmatrix}}$ for $j \in [0, w_2]$. As $\mathcal{B}$ knows $\boldsymbol{\alpha}$, $\mathbf{H}_1, \ldots, \mathbf{H}_{n+2}$, it can compute all components of ciphertext.

Then $\mathcal{B}$ computes $\mathbf{C}^*$ as it knows $\boldsymbol{\alpha}$, $\mathbf{H}_1, \ldots, \mathbf{H}_{n+2}$. Runs $(vk^*, sk^*) \leftarrow \mathsf{OTS.Gen}(1^\lambda)$. Then it evaluates $\xi^* = \mathcal{H}(\mathbf{C}^*, vk^*)$ to compute $\overline{\mathbf{C}}^* = (\overline{\mathbf{C}}_0^*, \mathbf{C}^*, vk^*, \sigma^*)$ where $\overline{\mathbf{C}}_0^* = g_1^{(\xi^*\mathbf{H}_{n+1} + \mathbf{H}_{n+2})\mathbf{B}\begin{pmatrix}\mathbf{s}_0\\ \hat{s}_0\end{pmatrix}}$ and $\sigma^* \leftarrow \mathsf{OTS.Sign}(sk^*, \overline{\mathbf{C}}_0)$. It outputs $\overline{\mathbf{C}}^*$.

Key Queries. Same as Phase-I secret key queries.

Dec Queries. Same as Phase-I decryption queries.

Guess. $\mathcal{A}$ halts with output $\mathfrak{b}'$. $\mathcal{B}$ outputs 1 if $\mathfrak{b}' = \mathfrak{b}$ and 0 otherwise.

B.2 Normal to Type-1 Key in Phase-I

Lemma 6 ($\mathsf{Game}_{2,i-1,3}$ **to** $\mathsf{Game}_{2,i,1}$)**.** *For $i = 1, \ldots, q_1$, for any efficient adversary $\mathcal{A}$ that makes at most q_1 pre-challenge key queries, at most q_2 post-challenge key queries and at most $q_{_D}$ decryption queries, there exists a PPT algorithm $\mathcal{B}$ such that $|\mathsf{Adv}^{2,i-1,3}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{2,i,1}_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\mathcal{D}_d\text{-MatDH}}_{\mathcal{B}}(\lambda)$.*

Proof. The algorithm $\mathcal{B}$ gets as input $(\mathbf{G}, g_2^{\mathbf{T}}, g_2^{\mathbf{T}\begin{pmatrix}\boldsymbol{y}\\ \hat{y}\end{pmatrix}})$ as $\mathcal{D}_d$-MatDH problem instance where $\hat{y} = 0$ or $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$, $\boldsymbol{y} \xleftarrow{\$} \mathbb{Z}_p^d$.

Setup. $\mathcal{B}$ chooses $\tilde{\mathbf{B}} \xleftarrow{\$} \mathbb{GL}_{p,d+1}, \mathbf{J} \xleftarrow{\$} \mathbb{GL}_{p,d}$ and sets $\mathbf{B} = \tilde{\mathbf{B}} \begin{pmatrix} \mathbf{I}_d & \mathbf{M}^{-\top}\mathbf{c}^{\top} \\ \mathbf{0} & -1 \end{pmatrix}$ and $\mathbf{D} = \begin{pmatrix} \mathbf{MJ} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix}$ where $\mathbf{T} = \begin{pmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{c} & 1 \end{pmatrix}$ due to $\mathcal{D}_d$-MatDH assumption. Then $\mathbf{Z} = \mathbf{B}^{-\top}\mathbf{D} = \tilde{\mathbf{B}}^{-\top} \begin{pmatrix} \mathbf{I}_d & \mathbf{0} \\ \mathbf{cM}^{-1} & -1 \end{pmatrix} \begin{pmatrix} \mathbf{MJ} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} = \tilde{\mathbf{B}}^{-\top}\mathbf{T} \begin{pmatrix} \mathbf{J} & \mathbf{0} \\ \mathbf{0} & -1 \end{pmatrix}$. Then define $\tilde{\mathbf{Z}} = \begin{pmatrix} \mathbf{J} & \mathbf{0} \\ \mathbf{0} & -1 \end{pmatrix}$ so that $\mathbf{Z} = \tilde{\mathbf{B}}^{-\top}\mathbf{T}\tilde{\mathbf{Z}}$. $\mathcal{B}$ therefore can compute the public parameters as $g_1^{\mathbf{B}\left(\begin{smallmatrix}\mathbf{I}_d \\ 0\end{smallmatrix}\right)} = g_1^{\tilde{\mathbf{B}}\left(\begin{smallmatrix}\mathbf{I}_d \\ 0\end{smallmatrix}\right)}$ and $g_2^{\mathbf{Z}} = g_2^{\tilde{\mathbf{B}}^{-\top}\mathbf{T}\tilde{\mathbf{Z}}}$. Then $\mathcal{B}$ chooses $\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)}$ and $\mathbb{H} = (\mathbf{H}_1, \ldots, \mathbf{H}_{n+2}) \xleftarrow{\$} \left(\mathbb{Z}_p^{(d+1)\times(d+1)}\right)^{(n+2)}$ and publishes public key mpk. Note that $\mathcal{B}$ cannot compute $\widehat{mpk}_b$ but can compute $\widehat{mpk}_z$ as it can compute $\widehat{mpk}_{base}$.

Key Queries. On j^{th} secret key query x ($j \le q_1$), outputs secret key $\mathbf{K}$ as follows.

- If $j > i$, $\mathcal{B}$ generates normal key $\mathsf{KeyGen}(x, msk)$.
- If $j < i$, $\mathcal{B}$ generates type-3 key $\mathsf{SFKeyGen}(x, msk, -, \widehat{mpk}_{base}, 3, \beta_j)$ for $\beta_j \xleftarrow{\$} \mathbb{Z}_p$.
- If $j = i$, $\mathcal{B}$ runs $(\mathbf{k} = (k_1, \ldots, k_{m_1}); m_2) \leftarrow \mathsf{EncK}(x, N)$. It uses random self-reducibility of Matrix-DH assumption to obtain $(\mathbf{G}, g_2^{\mathbf{T}}, g_2^{\mathbf{T}\left(\begin{smallmatrix}\mathbf{Y} \\ \hat{\boldsymbol{y}}\end{smallmatrix}\right)})$. The decision problem is now to find if $\hat{\boldsymbol{y}} = \mathbf{0}$ or $\hat{\boldsymbol{y}} \xleftarrow{\$} \left(\mathbb{Z}_p\right)^{m_2}$ where $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$ and $\mathbf{Y} \xleftarrow{\$} \left(\mathbb{Z}_p^d\right)^{m_2}$. $\mathcal{B}$ implicitly sets $\tilde{\mathbf{Z}}^{-1}\left(\begin{smallmatrix}\mathbf{Y} \\ \hat{\boldsymbol{y}}\end{smallmatrix}\right) = \mathbf{R} + \hat{\mathbf{R}} = \begin{pmatrix} \mathbf{r}_1 & \cdots & \mathbf{r}_{m_2} \\ \hat{r}_1 & \cdots & \hat{r}_{m_2} \end{pmatrix}$. Therefore $g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathbf{r}_j \\ \hat{r}_j\end{smallmatrix}\right)} = g_2^{\left(\tilde{\mathbf{B}}^{-\top}\mathbf{T}\tilde{\mathbf{Z}}\right)\left(\tilde{\mathbf{Z}}^{-1}\left(\begin{smallmatrix}\boldsymbol{y}_j \\ \hat{y}_j\end{smallmatrix}\right)\right)}$. As $\mathcal{B}$ has $g_2^{\mathbf{T}\left(\begin{smallmatrix}\mathbf{Y} \\ \hat{\boldsymbol{y}}\end{smallmatrix}\right)}$, $\boldsymbol{\alpha}$, $\tilde{\mathbf{B}}$, $\mathbf{H}_1, \ldots, \mathbf{H}_{n+2}$, it can compute all components of secret key. It is evident from the description that if $\hat{\boldsymbol{y}} = \mathbf{0}$, the key is a normal key whereas if $\hat{\boldsymbol{y}} \xleftarrow{\$} \left(\mathbb{Z}_p\right)^{m_2}$, the key is type-1 key.

Dec Queries. On decryption query $(x, \overline{\mathbf{C}})$ where $\overline{\mathbf{C}}$ is a ciphertext on data-index y, if the signature σ is not verified or if $R(x, y) \neq 1$, aborts. Otherwise $\mathcal{B}$ computes normal altKey $\hat{\mathbf{K}} \leftarrow \mathsf{AltKeyGen}(\overline{\mathbf{C}}, x, msk)$ and returns $\mathsf{AltDec}(\overline{\mathbf{C}}, \hat{\mathbf{K}})$ to $\mathcal{A}$.

Challenge. On receiving the challenge (y^*, M_0, M_1), $\mathcal{B}$ picks $\mathfrak{b} \xleftarrow{\$} \{0, 1\}$. It runs $(\mathbf{c} = (c_1, \ldots, c_{w_1}); w_2) \leftarrow \mathsf{EncC}(y^*, N)$ and for $j \in [0, w_2]$ chooses $\left(\begin{smallmatrix}\mathbf{s}'_j \\ \hat{s}'_j\end{smallmatrix}\right) \xleftarrow{\$} \mathbb{Z}_p^{(d+1)}$ and implicitly sets $\left(\begin{smallmatrix}\mathbf{s}_j \\ \hat{s}_j\end{smallmatrix}\right) = \mathbf{B}^{-1}\left(\begin{smallmatrix}\mathbf{s}'_j \\ \hat{s}'_j\end{smallmatrix}\right)$.

Then $\mathcal{B}$ computes $\mathbf{C}^*$ as it knows $\boldsymbol{\alpha}$, $\mathbf{H}_1, \ldots, \mathbf{H}_{n+2}$. Runs $(vk^*, sk^*) \leftarrow \mathsf{OTS.Gen}(1^\lambda)$. Then it evaluates $\xi^* = \mathcal{H}(\mathbf{C}^*, vk^*)$ to compute $\overline{\mathbf{C}}^* = (\overline{\mathbf{C}}_0^*, \mathbf{C}^*, vk^*, \sigma^*)$ where $\overline{\mathbf{C}}_0^* = g_1^{(\xi^*\mathbf{H}_{n+1} + \mathbf{H}_{n+2})\left(\begin{smallmatrix}\mathbf{s}'_0 \\ \hat{s}'_0\end{smallmatrix}\right)}$ and $\sigma^* \leftarrow \mathsf{OTS.Sign}(sk^*, \overline{\mathbf{C}}_0)$. It outputs $\overline{\mathbf{C}}^*$.

Key Queries. On j^{th} secret key query x ($j \in [q_1 + 1, q]$), $\mathcal{B}$ generates normal key $\mathsf{KeyGen}(x, msk)$.

Dec Queries. Same as Phase-I decryption queries.

Guess. $\mathcal{A}$ halts with output $\mathfrak{b}'$. $\mathcal{B}$ outputs 1 if $\mathfrak{b}' = \mathfrak{b}$ and 0 otherwise.

B.3 Randomizing via Parameter Hiding

Here we modify $\mathsf{SFSetup}$ to define setup algorithm $\mathsf{SFSetup}'$ to introduce some *extra randomness* in the semi-functional components of $\widehat{mpk}_b$ and $\widehat{mpk}_z$. We also describe the consequence of such newly introduced randomness in the outputs of SFEnc, $\mathsf{SFKeyGen}$ and $\mathsf{SFAltKeyGen}$.

- $\mathsf{SFSetup}'(1^\lambda, \kappa)$: It outputs $mpk, msk, \widehat{mpk}_{base}$ in exactly the same way. It additionally chooses $\hat{\mathbf{h}} = (\hat{h}_1, \ldots, \hat{h}_{n+2}) \xleftarrow{\$} \mathbb{Z}_p^{n+2}$ and computes $\widehat{mpk}_b = \left(e(g_1, g_2)^{\boldsymbol{\alpha}^\top \mathbf{B}\left(\begin{smallmatrix}\mathbf{0} \\ 1\end{smallmatrix}\right)}, g_1^{\mathbf{B}\left(\begin{smallmatrix}\mathbf{0} \\ 1\end{smallmatrix}\right)}, g_1^{\mathbf{H}_1\mathbf{B}\left(\begin{smallmatrix}\mathbf{0} \\ 1\end{smallmatrix}\right) + \mathbf{B}\left(\begin{smallmatrix}\mathbf{0} \\ \hat{h}_1\end{smallmatrix}\right)}, \ldots, g_1^{\mathbf{H}_{n+2}\mathbf{B}\left(\begin{smallmatrix}\mathbf{0} \\ 1\end{smallmatrix}\right) + \mathbf{B}\left(\begin{smallmatrix}\mathbf{0} \\ \hat{h}_{n+2}\end{smallmatrix}\right)} \right)$

and $\widehat{mpk}_z = \left(g_2^{\mathbf{H}_1^\top \mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\1\end{smallmatrix}\right) + \mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\\hat{h}_1\end{smallmatrix}\right)}, \ldots, g_2^{\mathbf{H}_{n+2}^\top \mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\1\end{smallmatrix}\right) + \mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\\hat{h}_{n+2}\end{smallmatrix}\right)} \right)$.

– $\mathsf{SFKeyGen}(x, msk, \widehat{mpk}_z, \widehat{mpk}_{base}, \text{type}, \beta)$: Runs $(\mathbf{k}; m_2) \leftarrow \mathsf{EncK}(x, N)$. Chooses $\mathbf{r}_1, \ldots, \mathbf{r}_{m_2} \xleftarrow{\$} \mathbb{Z}_p^d$ and $\hat{r}_1, \ldots, \hat{r}_{m_2} \xleftarrow{\$} \mathbb{Z}_p$. Then it defines $\mathbf{R} = \left(\left(\begin{smallmatrix}\mathbf{r}_1\\0\end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix}\mathbf{r}_{m_2}\\0\end{smallmatrix}\right) \right) \in \left(\mathbb{Z}_p^{(d+1)}\right)^{m_2}$ and $\widehat{\mathbf{R}} = \left(\left(\begin{smallmatrix}\mathbf{0}\\\hat{r}_1\end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix}\mathbf{0}\\\hat{r}_{m_2}\end{smallmatrix}\right) \right) \in \left(\mathbb{Z}_p^{(d+1)}\right)^{m_2}$.
Outputs the secret key

$$\mathbf{K} = \begin{cases} g_2^{\mathbf{k}(\boldsymbol{\alpha}, \mathbf{R}, \mathbb{H}) + \mathbf{k}(\mathbf{0}, \widehat{\mathbf{R}}, \mathbb{H}, \hat{\mathbf{h}})} & \text{if type} = 1 \\ g_2^{\mathbf{k}(\boldsymbol{\alpha}, \mathbf{R}, \mathbb{H}) + \mathbf{k}\left(\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\\beta\end{smallmatrix}\right), \widehat{\mathbf{R}}, \mathbb{H}, \hat{\mathbf{h}}\right)} & \text{if type} = 2 \end{cases}$$

where $\mathbf{k}(\boldsymbol{\alpha}, \mathbf{R}, \mathbb{H}) + \mathbf{k}(\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\\beta\end{smallmatrix}\right), \widehat{\mathbf{R}}, \mathbb{H}, \hat{\mathbf{h}}) =$

$$\left\{ b_\iota \boldsymbol{\alpha} + b_\iota \mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\\beta\end{smallmatrix}\right) + \sum_{j \in [m_2]} b_{\iota j} \mathbf{Z}\left(\begin{smallmatrix}\mathbf{r}_j\\\hat{r}_j\end{smallmatrix}\right) + \sum_{\substack{j \in [m_2] \\ k \in [n]}} b_{\iota jk} \left(\mathbf{H}_k^\top \mathbf{Z}\left(\begin{smallmatrix}\mathbf{r}_j\\\hat{r}_j\end{smallmatrix}\right) + \mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\\hat{h}_k \hat{r}_j\end{smallmatrix}\right) \right) \right\}_{\iota \in [m_1]}.$$

– $\mathsf{SFEnc}(y, M, mpk, \widehat{mpk}_b)$: It runs $(\mathbf{c}; w_2) \leftarrow \mathsf{EncC}(y, N)$. Chooses $\mathbf{s}_0, \ldots, \mathbf{s}_{w_2} \xleftarrow{\$} \mathbb{Z}_p^d$ and $\hat{s}_0, \ldots, \hat{s}_{w_2} \xleftarrow{\$} \mathbb{Z}_p$. It defines $\mathbf{S} = \left(\left(\begin{smallmatrix}\mathbf{s}_0\\0\end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix}\mathbf{s}_{w_2}\\0\end{smallmatrix}\right) \right) \in \left(\mathbb{Z}_p^{(d+1)}\right)^{(w_2+1)}$ and $\widehat{\mathbf{S}} = \left(\left(\begin{smallmatrix}\mathbf{0}\\\hat{s}_0\end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix}\mathbf{0}\\\hat{s}_{w_2}\end{smallmatrix}\right) \right) \in \left(\mathbb{Z}_p^{(d+1)}\right)^{(w_2+1)}$. The semi-functional ciphertext it computes is $\mathbf{C} = (\mathbf{C}_1, \ldots, \mathbf{C}_{w_1}, \mathbf{C}_{w_1+1})$ where

$$\mathbf{C}_{\tilde{\iota}} = g_1^{\mathbf{c}_{\tilde{\iota}}(\mathbf{S}, \mathbb{H}) + \mathbf{c}_{\tilde{\iota}}(\widehat{\mathbf{S}}, \mathbb{H}, \hat{\mathbf{h}})} = g_1^{\left(\sum_{j \in [0, w_2]} a_{\tilde{\iota} j} \mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_j\\\hat{s}_j\end{smallmatrix}\right) + \sum_{\substack{j \in [0, w_2] \\ k \in [n]}} a_{\tilde{\iota} jk} \left(\mathbf{H}_k \mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_j\\\hat{s}_j\end{smallmatrix}\right) + \mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\\hat{h}_k \hat{s}_j\end{smallmatrix}\right) \right) \right)}$$

for $\tilde{\iota} \in [w_1]$ and $\mathbf{C}_{w_1+1} = M \cdot e(g_1, g_2)^{\boldsymbol{\alpha}^\top \mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\\hat{s}_0\end{smallmatrix}\right)}$.
It outputs $\overline{\mathbf{C}} = (\overline{\mathbf{C}}_0, \mathbf{C}, vk, \sigma)$ where $\overline{\mathbf{C}}_0 = g_1^{(\xi \mathbf{H}_{n+1} + \mathbf{H}_{n+2}) \mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\\hat{s}_0\end{smallmatrix}\right) + \mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\(\xi \hat{h}_{n+1} + \hat{h}_{n+2}) \hat{s}_0\end{smallmatrix}\right)}$ such that $\xi = \mathcal{H}(\mathbf{C}, vk)$ and $\sigma = \mathsf{OTS.Sign}(sk, \overline{\mathbf{C}}_0)$ for $(vk, sk) \leftarrow \mathsf{OTS.Gen}(1^\lambda)$.

– $\mathsf{SFAltKeyGen}(\overline{\mathbf{C}}, x, msk, \widehat{mpk}_z, \widehat{mpk}_{base}, \text{type}, \eta)$: Runs $(\mathbf{k}; m_2) \leftarrow \mathsf{EncK}(x, N)$ and $\mathbf{E} \leftarrow \mathsf{Pair}(x, y)$. Chooses $\mathbf{r}_1, \ldots, \mathbf{r}_{m_2}, \mathfrak{r} \xleftarrow{\$} \mathbb{Z}_p^d$ and $\hat{\mathfrak{r}} \xleftarrow{\$} \mathbb{Z}_p$. Then it defines $\mathbf{R} = \left(\left(\begin{smallmatrix}\mathbf{r}_1\\0\end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix}\mathbf{r}_{m_2}\\0\end{smallmatrix}\right) \right) \in \left(\mathbb{Z}_p^{(d+1)}\right)^{(m_2+1)}$.
Then the normal key $\mathbf{K} = \left\{ g_2^{k_\iota(\boldsymbol{\alpha}, \mathbf{R}, \mathbb{H})} \right\}_{\iota \in [m_1]} \in \left(\mathbb{G}_2^{(d+1)}\right)^{m_1}$ where each

$$k_\iota(\boldsymbol{\alpha}, \mathbf{R}, \mathbb{H}) = b_\iota \boldsymbol{\alpha} + \sum_{j \in [m_2]} b_{\iota j} \mathbf{Z}\left(\begin{smallmatrix}\mathbf{r}_j\\0\end{smallmatrix}\right) + \sum_{\substack{j \in [m_2] \\ k \in [n]}} b_{\iota jk} \mathbf{H}_k^\top \mathbf{Z}\left(\begin{smallmatrix}\mathbf{r}_j\\0\end{smallmatrix}\right) \text{ for } \iota \in [m_1]$$

Then it computes $(\tilde{K}_1, \ldots, \tilde{K}_{w_1})$ where $\tilde{K}_{\tilde{\iota}} = \prod_{\iota \in [m_1]} (\mathbf{K}_\iota)^{\mathbf{E}_{\iota \tilde{\iota}}}$ for each $\tilde{\iota} \in [w_1]$.
Defines modified key $\hat{\mathbf{K}} = (K_0, \Phi \cdot \tilde{K}_1, \tilde{K}_2, \ldots, \tilde{K}_{w_1})$ where

$$(K_0, \Phi) = \begin{cases} \left(g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\\hat{\mathfrak{r}}\end{smallmatrix}\right)}, g_2^{(\xi \mathbf{H}_{n+1}^\top + \mathbf{H}_{n+2}^\top) \mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\\hat{\mathfrak{r}}\end{smallmatrix}\right) + \mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\(\xi \hat{h}_{n+1} + \hat{h}_{n+2}) \hat{\mathfrak{r}}\end{smallmatrix}\right)} \right) & \text{if type} = 1 \\ \left(g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\\hat{\mathfrak{r}}\end{smallmatrix}\right)}, g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\\eta u\end{smallmatrix}\right) + (\xi \mathbf{H}_{n+1}^\top + \mathbf{H}_{n+2}^\top) \mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\\hat{\mathfrak{r}}\end{smallmatrix}\right) + \mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\(\xi \hat{h}_{n+1} + \hat{h}_{n+2}) \hat{\mathfrak{r}}\end{smallmatrix}\right)} \right) & \text{if type} = 2, \end{cases}$$

$u = \sum_{\iota \in [m_1]} b_\iota \mathbf{E}_{\iota 1}$ and $\xi = \mathcal{H}(\mathbf{C}, vk)$ for the given $\overline{\mathbf{C}} = (\overline{\mathbf{C}}_0, \mathbf{C}, vk, \sigma)$.

We here show that outputs of $\mathsf{SFSetup}$ and $\mathsf{SFSetup}'$ are identically distributed. This allows us to replace $\mathsf{SFSetup}$ by $\mathsf{SFSetup}'$ and run $\mathsf{SFKeyGen}$, SFEnc and $\mathsf{SFAltKeyGen}$ to generate the secret keys, the challenge ciphertext and the altKeys containing the randomness newly introduced via $\hat{\mathbf{h}}$. This result will be used in arguing indistinguishibility of type-1 and type-2 keys of both secret keys and altKeys (Lemma 2, Lemma 7 and Lemma 10).

Claim. The outputs of $\mathsf{SFSetup}$ and $\mathsf{SFSetup}'$ are identically distributed.

Proof. Due to parameter-hiding lemma in Section 2.3, $\mathbf{H}_i \xleftarrow{\$} \mathbb{Z}_p^{(d+1)\times(d+1)}$, $\mathbf{B} \xleftarrow{\$} \mathbb{GL}_{p,d+1}$ and $\hat{h}_i \xleftarrow{\$} \mathbb{Z}_p$, both $R_{\mathbf{H}_i,\hat{h}_i}$ and $\mathbf{H}_i$ are identically distributed where $R_{\mathbf{H}_i,\hat{h}_i} = \mathbf{H}_i + \mathbf{B}\begin{pmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \hat{h}_i\end{pmatrix}\mathbf{B}^{-1}$ for $i \in [n+2]$. It can easily be verified that for each $i \in [n+2]$, $R_{\mathbf{H}_i,\hat{h}_i}\mathbf{B}\left(\begin{smallmatrix}\mathbf{I}_d\\ 0\end{smallmatrix}\right) = \mathbf{H}_i\mathbf{B}\left(\begin{smallmatrix}\mathbf{I}_d\\ 0\end{smallmatrix}\right)$, $R^{\top}_{\mathbf{H}_i,\hat{h}_i}\mathbf{Z}\left(\begin{smallmatrix}\mathbf{I}_d\\ 0\end{smallmatrix}\right) = \mathbf{H}_i^{\top}\mathbf{Z}\left(\begin{smallmatrix}\mathbf{I}_d\\ 0\end{smallmatrix}\right)$, $R_{\mathbf{H}_i,\hat{h}_i}\mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\ 1\end{smallmatrix}\right) = \mathbf{H}_i\mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\ 1\end{smallmatrix}\right) + \mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\ \hat{h}_i\end{smallmatrix}\right)$ and $R^{\top}_{\mathbf{H}_i,\hat{h}_i}\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ 1\end{smallmatrix}\right) = \mathbf{H}_i^{\top}\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ 1\end{smallmatrix}\right) + \mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ \hat{h}_i\end{smallmatrix}\right)$. Note that this replacement doesn't change mpk and is therefore oblivious to any adversary. Only the description of $\widehat{mpk}_b$ and $\widehat{mpk}_z$ of $\mathsf{SFSetup}'$ gets modified. It is evident that this change does not affect neither the normal nor the type-3 semi-functional forms of secret keys and altKeys.

B.4 Type-1 to Type-2 Key in Phase-I

Lemma 7 ($\mathsf{Game}_{2,i,1}$ to $\mathsf{Game}_{2,i,2}$). *For $i = 1, \ldots, q_1$, for any efficient adversary $\mathcal{A}$ that makes at most q_1 pre-challenge key queries, at most q_2 post-challenge key queries and at most $q_{_D}$ decryption queries, there exists a PPT algorithm $\mathcal{B}$ such that $|\mathsf{Adv}^{2,i,1}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{2,i,2}_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\mathrm{CMH}}_{\mathcal{B}}(\lambda)$.*

Proof. In this co-selective security game of pair encoding scheme, the algorithm $\mathcal{B}$ gets as input the group description $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$.

Setup. $\mathcal{B}$ chooses $\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)}$, $\mathbb{H} = (\mathbf{H}_1, \ldots, \mathbf{H}_{n+2}) \xleftarrow{\$} \left(\mathbb{Z}_p^{(d+1)\times(d+1)}\right)^{(n+2)}$, $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)\times(d+1)}$, $\tilde{\mathbf{D}} \xleftarrow{\$} \mathbb{GL}_{p,d}$ and defines $\mathbf{D} = \begin{pmatrix}\tilde{\mathbf{D}} & \mathbf{0}\\ \mathbf{0} & 1\end{pmatrix}$ and $\mathbf{Z} = \mathbf{B}^{-\top}\mathbf{D}$. It computes mpk, $\widehat{mpk}_{base}$ and msk. We note that these elements are distributed as if they are output of $\mathsf{SFSetup}'$.

Key Queries. On j^{th} secret key query x ($j \leq q_1$), outputs secret key $\mathbf{K}$ as follows.

- If $j > i$, $\mathcal{B}$ generates normal key $\mathsf{KeyGen}(x, msk)$.
- If $j < i$, $\mathcal{B}$ generates type-3 key $\mathsf{SFKeyGen}(x, msk, -, \widehat{mpk}_{base}, 3, \beta_j)$ after choosing $\beta_j \xleftarrow{\$} \mathbb{Z}_p$.
- If $j = i$, $\mathcal{B}$ forwards x as the challenge query to the challenger to receive $\mathbf{V} = g_2^{\mathbf{k}(\beta,\hat{\mathbf{r}},\hat{\mathbf{h}})}$ where $(\mathbf{k} = (k_1, \ldots, k_{m_1}); m_2) \leftarrow \mathsf{EncK}(x, N)$. $\mathcal{B}$ has to decide if $\beta = 0$ or $\beta \xleftarrow{\$} \mathbb{Z}_p$. It is to be noted that $\hat{\mathbf{r}}$ and $\hat{\mathbf{h}}$ are chosen by the challenger of CMH-security game, unknown to $\mathcal{B}$. Now $\mathcal{B}$ computes the normal part of the key by computing $\left\{g_2^{k_\iota(\boldsymbol{\alpha},\mathbf{R},\mathbb{H})}\right\}_{\iota\in[m_1]}$ for $\mathbf{R} = \left(\left(\begin{smallmatrix}\mathbf{r}_1\\ 0\end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix}\mathbf{r}_{m_2}\\ 0\end{smallmatrix}\right)\right)$ such that $\mathbf{r}_1, \ldots, \mathbf{r}_{m_2} \xleftarrow{\$} \mathbb{Z}_p^d$. To compute the semi-functional part, that contains $\hat{\mathbf{h}}$ which is unknown to $\mathcal{B}$, it implicitly sets $\hat{\mathbf{R}} = \left(\left(\begin{smallmatrix}\mathbf{0}\\ \hat{r}_1\end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix}\mathbf{0}\\ \hat{r}_{m_2}\end{smallmatrix}\right)\right)$ where $\hat{\mathbf{r}} = (\hat{r}_1, \ldots, \hat{r}_{m_2})$. Then the semi-functional component of the key is
$$g_2^{\mathbf{k}'(\beta_i,\hat{\mathbf{R}},\mathbb{H},\hat{\mathbf{h}})} = \left\{ g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ k_\iota(\beta,\hat{\mathbf{r}},\hat{\mathbf{h}})\end{smallmatrix}\right)} \prod_{\substack{j\in[m_2]\\ k\in[n]}} g_2^{b_{\iota jk}\mathbf{H}_k^{\top}\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ \hat{r}_j\end{smallmatrix}\right)} \right\}_{\iota\in[m_1]}.$$
Notice that $\mathcal{B}$ implicitly sets β_i to be β that is actually set by the challenger of CMH-security game and unknown to $\mathcal{B}$. Since $\mathcal{B}$ already have received $\mathbf{V} = (V_1, \ldots, V_{m_1})$ from the challenger of CMH-security game, it uses V_ι to compute the first component of the right hand side of the above equation i.e. $g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ k_\iota(\beta,\hat{\mathbf{r}},\hat{\mathbf{h}})\end{smallmatrix}\right)} = V_\iota^{\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ 1\end{smallmatrix}\right)}$ for $\iota \in [m_1]$.

However to compute the second component $\left(\prod_{\substack{j\in[m_2]\\k\in[n]}} g_2^{b_{\iota jk}\mathbf{H}_k^\top \mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ \hat{r}_j\end{smallmatrix}\right)}\right)$ of the semi-functional part of the secret key, $\mathcal{B}$ needs to know $\hat{\mathbf{r}} = (\hat{r}_1, \ldots, \hat{r}_{m_2})$. For each of $j \in [m_2]$, two cases can happen.

- Either there is $\iota' \in [m_1]$ such that $k_{\iota'}(\beta, \hat{\mathbf{r}}, \hat{\mathbf{h}}) = \hat{r}_j$, that lets $\mathcal{B}$ to know $\hat{r}_j$.
- Or there is no such $\iota' \in [m_1]$ for which $k_{\iota'}(\beta, \hat{\mathbf{r}}, \hat{\mathbf{h}}) = \hat{r}_j$. Then due to regular decryption properties of pair encoding (precisely Property $\mathcal{P}2$), $b_{\iota'' jk} = 0$ for all $\iota'' \in [m_1]$, $k \in [n]$.

$\mathcal{B}$ uses the normal part of the key and the semi-functional part of the key to generate the secret key and hands it over to $\mathcal{A}$.

Dec Queries. On decryption query $(x, \overline{\mathbf{C}})$ where $\overline{\mathbf{C}}$ is a ciphertext on data-index y, if the signature σ is not verified or if $R(x, y) \neq 1$, aborts. Otherwise $\mathcal{B}$ computes normal altKey $\hat{\mathbf{K}} \leftarrow \mathsf{AltKeyGen}(\overline{\mathbf{C}}, x, msk)$ and returns $\mathsf{AltDec}(\overline{\mathbf{C}}, \hat{\mathbf{K}})$ to $\mathcal{A}$.

Challenge. On receiving the challenge (y^*, M_0, M_1), $\mathcal{B}$ picks $\mathfrak{b} \xleftarrow{\$} \{0, 1\}$. It makes the ciphertext query on y^* to the challenger of CMH-security game. It is possible to make such a challenge query as $R(x, y^*) = 0$ for all key queries. $\mathcal{B}$ receives $\mathbf{U} \leftarrow g_1^{\mathbf{c}(\hat{\mathbf{s}}, \hat{\mathbf{h}})}$.

$\mathcal{B}$ first computes the normal part of the ciphertext by computing $g_1^{c_{\tilde{\iota}}(\mathbf{S}, \mathbb{H})}$ for $\mathbf{S} = \left(\left(\begin{smallmatrix}\mathbf{s}_0\\ 0\end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix}\mathbf{s}_{w_2}\\ 0\end{smallmatrix}\right)\right)$ such that $\mathbf{s}_0, \ldots, \mathbf{s}_{w_2} \xleftarrow{\$} \mathbb{Z}_p^d$.

To compute the semi-functional part, that contains $\hat{\mathbf{h}}$ which is unknown to $\mathcal{B}$, it implicitly sets $\hat{\mathbf{S}} = \left(\left(\begin{smallmatrix}\mathbf{0}\\ \hat{s}_0\end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix}\mathbf{0}\\ \hat{s}_{w_2}\end{smallmatrix}\right)\right)$ where $\hat{\mathbf{s}} = (\hat{s}_0, \ldots, \hat{s}_{w_2})$. Then it computes the semi-functional component of the ciphertext as

$$g_1^{\mathbf{c}'(\hat{\mathbf{S}}, \mathbb{H}, \hat{\mathbf{h}})} = \left\{ g_1^{\mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\ c_{\tilde{\iota}}(\hat{\mathbf{s}}, \hat{\mathbf{h}})\end{smallmatrix}\right)} \prod_{\substack{j\in[0,w_2]\\k\in[n]}} g_1^{a_{\tilde{\iota} jk}\mathbf{H}_k\mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\ \hat{s}_j\end{smallmatrix}\right)} \right\}_{\tilde{\iota}\in[w_1]}.$$

Since $\mathcal{B}$ already have received $\mathbf{U} = (U_1, \ldots, U_{w_1})$ from the challenger of CMH-security game, it uses $U_{\tilde{\iota}}$ to compute the first component of the right hand side of the above equation i.e. $g_1^{\mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\ c_{\tilde{\iota}}(\hat{\mathbf{s}}, \hat{\mathbf{h}})\end{smallmatrix}\right)} = U_{\tilde{\iota}}^{\mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\ 1\end{smallmatrix}\right)}$.

However to compute the second component $\left(\prod_{\substack{j\in[0,w_2]\\k\in[n]}} g_1^{a_{\tilde{\iota} jk}\mathbf{H}_k\mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\ \hat{s}_j\end{smallmatrix}\right)}\right)$ of the semi-functional part of the ciphertext, $\mathcal{B}$ needs to know $\hat{\mathbf{s}} = (\hat{s}_0, \ldots, \hat{s}_{w_2})$. For each of $j \in [0, w_2]$, two cases can happen.

- Either there is $\tilde{\iota}' \in [w_1]$ such that $c_{\tilde{\iota}'}(\hat{\mathbf{s}}, \hat{\mathbf{h}}) = \hat{s}_j$, that lets $\mathcal{B}$ to know $\hat{s}_j$.
- Or there is no such $\tilde{\iota}' \in [w_1]$ for which $c_{\tilde{\iota}'}(\hat{\mathbf{s}}, \hat{\mathbf{h}}) = \hat{s}_j$. Then due to regular decryption properties of pair encoding (precisely Property $\mathcal{P}3$), $a_{\tilde{\iota}'' jk} = 0$ for all $\tilde{\iota}'' \in [w_1]$, $k \in [n]$.

Due to regular decryption properties of pair encoding (precisely Property $\mathcal{P}4$), $\mathcal{B}$ also can compute the semi-functional component of the blinding factor $e(g_1^{\boldsymbol{\alpha}^\top \mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\ \hat{s}_0\end{smallmatrix}\right)}, g_2)$ as $g_1^{\hat{s}_0}$ is available in $\mathbf{U}$.

$\mathcal{B}$ uses the normal part of the ciphertext and the semi-functional part of the ciphertext to generate the $\mathbf{C}^*$. Runs $(vk^*, sk^*) \leftarrow \mathsf{OTS.Gen}(1^\lambda)$. Then it evaluates $\xi^* = \mathcal{H}(\mathbf{C}^*, vk^*)$ to compute $\overline{\mathbf{C}}_0^*$ and defines $\overline{\mathbf{C}}^* = (\overline{\mathbf{C}}_0^*, \mathbf{C}^*, vk^*, \sigma^*)$ for $\sigma^* \leftarrow \mathsf{OTS.Sign}(sk^*, \overline{\mathbf{C}}_0)$. It outputs $\overline{\mathbf{C}}^*$.

Key Queries. On j^{th} secret key query x ($j \in [q_1 + 1, q]$), $\mathcal{B}$ generates secret key $\mathbf{K} \leftarrow \mathsf{KeyGen}(x, msk)$.

Dec Queries. Same as Phase-I decryption queries.

Guess. $\mathcal{A}$ halts with output $\mathfrak{b}'$. $\mathcal{B}$ outputs 1 if $\mathfrak{b}' = \mathfrak{b}$ and 0 otherwise.

B.5 Type-2 to Type-3 Key in Phase-I

Lemma 8 ($\mathsf{Game}_{2,i,2}$ **to** $\mathsf{Game}_{2,i,3}$)**.** *For $i = 1, \ldots, q_1$, for any efficient adversary $\mathcal{A}$ that makes at most q_1 pre-challenge key queries, at most q_2 post-challenge key queries and at most $q_{_D}$ decryption queries, there exists a PPT algorithm $\mathcal{B}$ such that $|\mathsf{Adv}^{2,i,2}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{2,i,3}_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\mathcal{D}_d\text{-MatDH}}_{\mathcal{B}}(\lambda)$.*

Proof. The algorithm $\mathcal{B}$ gets as input $(\mathbf{G}, g_2^{\mathbf{T}}, g_2^{\mathbf{T}\begin{pmatrix}\boldsymbol{y}\\ \hat{y}\end{pmatrix}})$ as $\mathcal{D}_d$-MatDH problem instance where $\hat{y} = 0$ or $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$, $\boldsymbol{y} \xleftarrow{\$} \mathbb{Z}_p^d$.

The simulator description is same as Lemma 6 except while answering i^{th} query. For $\iota \in [m_2]$, each ι^{th} component of secret key of i^{th} key query is now multiplied by $g_2^{k_\iota(\mathbf{Z}\begin{pmatrix}\mathbf{0}\\ \beta_i\end{pmatrix}, \mathbf{0}, \mathbb{H})} \in \mathbb{G}_2^{(d+1)}$. As $\mathcal{B}$ knows $\widehat{mpk}_{\text{base}}$, it chooses $\beta_i \xleftarrow{\$} \mathbb{Z}_p$ to perform the simulation. In the similar light of Lemma 6, we see that if $\hat{\boldsymbol{y}} = \mathbf{0}$, the key is a type-3 key whereas if $\hat{\boldsymbol{y}} \xleftarrow{\$} \left(\mathbb{Z}_p\right)^{m_2}$, the key is type-2 key.

B.6 Normal to Type-1 Key in Phase-II

Lemma 9 ($\mathsf{Game}_{2,q_1,3}$ **to** $\mathsf{Game}_{2,q_1+1,1}$)**.** *For any efficient adversary $\mathcal{A}$ that makes at most q_1 pre-challenge key queries, at most q_2 post-challenge key queries and at most $q_{_D}$ decryption queries, there exists a PPT algorithm $\mathcal{B}$ such that $|\mathsf{Adv}^{2,q_1,3}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{2,q_1+1,1}_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\mathcal{D}_d\text{-MatDH}}_{\mathcal{B}}(\lambda)$.*

Proof. The algorithm $\mathcal{B}$ gets as input $(\mathbf{G}, g_2^{\mathbf{T}}, g_2^{\mathbf{T}\begin{pmatrix}\boldsymbol{y}\\ \hat{y}\end{pmatrix}})$ as $\mathcal{D}_d$-MatDH problem instance where $\hat{y} = 0$ or $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$, $\boldsymbol{y} \xleftarrow{\$} \mathbb{Z}_p^d$.

The simulator description is same as Lemma 6 except the simulator has to generate all post-challenge keys at once. Here the simulator again uses random self-reducibility property of Matrix-DH problem to create $q_2 m_2$ many instance of the given problem. It uses first m_2 instances to answer $(q_1+1)^{th}$ key query, next m_2 instances to answer $(q_1+2)^{th}$ key query, and so on. Similar to the proof of Lemma 6, we see that if $\hat{\boldsymbol{y}} = \mathbf{0}$, the key is a normal key whereas if $\hat{\boldsymbol{y}} \xleftarrow{\$} \left(\mathbb{Z}_p\right)^{m_2}$, the key is type-1 key.

B.7 Type-1 to Type-2 Key in Phase-II

Lemma 10 ($\mathsf{Game}_{2,q_1+1,1}$ **to** $\mathsf{Game}_{2,q_1+1,2}$)**.** *For any efficient adversary $\mathcal{A}$ that makes at most q_1 pre-challenge key queries, at most q_2 post-challenge key queries and at most $q_{_D}$ decryption queries, there exists a PPT algorithm $\mathcal{B}$ such that $|\mathsf{Adv}^{2,q_1+1,1}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{2,q_1+1,2}_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\text{SMH}}_{\mathcal{B}}(\lambda)$.*

Proof. In this selective security game of pair encoding scheme, the algorithm $\mathcal{B}$ gets as input the group description $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$.

Setup. $\mathcal{B}$ chooses $\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)}$ and $\mathbb{H} = (\mathbf{H}_1, \ldots, \mathbf{H}_{n+2}) \xleftarrow{\$} \left(\mathbb{Z}_p^{(d+1)\times(d+1)}\right)^{(n+2)}$, $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)\times(d+1)}$, $\tilde{\mathbf{D}} \xleftarrow{\$} \mathbb{GL}_{p,d}$ and defines $\mathbf{D} = \begin{pmatrix}\tilde{\mathbf{D}} & \mathbf{0}\\ \mathbf{0} & 1\end{pmatrix}$ and $\mathbf{Z} = \mathbf{B}^{-\top}\mathbf{D}$. It computes mpk and msk and gives mpk to $\mathcal{A}$.

Key Queries. On j^{th} secret key query x ($j \leq q_1$), $\mathcal{B}$ generates type-3 secret key $\mathbf{K} \leftarrow \mathsf{SFKeyGen}(x, msk, -, \widehat{mpk}_{base}, 3, \beta_j)$ after choosing $\beta_j \xleftarrow{\$} \mathbb{Z}_p$.

Dec Queries. On decryption query $(x, \overline{\mathbf{C}})$ where $\overline{\mathbf{C}}$ is a ciphertext on data-index y, if the signature σ is not verified or if $R(x, y) \neq 1$, aborts. Otherwise $\mathcal{B}$ computes normal altKey $\hat{\mathbf{K}} \leftarrow \mathsf{AltKeyGen}(\overline{\mathbf{C}}, x, msk)$ and returns $\mathsf{AltDec}(\overline{\mathbf{C}}, \hat{\mathbf{K}})$ to $\mathcal{A}$.

Challenge. On receiving the challenge (y^*, M_0, M_1), $\mathcal{B}$ picks $\mathfrak{b} \xleftarrow{\$} \{0,1\}$. It makes the challenge query on y^* to the challenger of SMH-security game. It is possible to make such a challenge query as $R(x, y^*) = 0$ for all key queries. $\mathcal{B}$ receives $\mathbf{U} \leftarrow g_1^{\mathbf{c}(\hat{\mathbf{s}},\hat{\mathbf{h}})}$.

$\mathcal{B}$ first computes the normal part of the ciphertext by computing $g_1^{c_{\tilde{\iota}}(\mathbf{S},\mathbb{H})}$ for $\mathbf{S} = \left(\begin{pmatrix}\mathbf{s}_0\\0\end{pmatrix}, \ldots, \begin{pmatrix}\mathbf{s}_{w_2}\\0\end{pmatrix}\right)$ such that $\mathbf{s}_0, \ldots, \mathbf{s}_{w_2} \xleftarrow{\$} \mathbb{Z}_p^d$.

To compute the semi-functional part, that contains $\hat{\mathbf{h}}$ which is unknown to $\mathcal{B}$, it implicitly sets $\hat{\mathbf{S}} = \left(\begin{pmatrix}\mathbf{0}\\\hat{s}_0\end{pmatrix}, \ldots, \begin{pmatrix}\mathbf{0}\\\hat{s}_{w_2}\end{pmatrix}\right)$ where $\hat{\mathbf{s}} = (\hat{s}_0, \ldots, \hat{s}_{w_2})$. Then it computes the semi-functional component of the ciphertext as

$$g_1^{\mathbf{c}'(\hat{\mathbf{S}},\mathbb{H},\hat{\mathbf{h}})} = \left\{ g_1^{\mathbf{B}\begin{pmatrix}\mathbf{0}\\c_{\tilde{\iota}}(\hat{\mathbf{s}},\hat{\mathbf{h}})\end{pmatrix}} \prod_{\substack{j\in[0,w_2]\\k\in[n]}} g_1^{a_{\tilde{\iota}jk}\mathbf{H}_k\mathbf{B}\begin{pmatrix}\mathbf{0}\\\hat{s}_j\end{pmatrix}} \right\}_{\tilde{\iota}\in[w_1]}.$$

Since $\mathcal{B}$ already has received $\mathbf{U} = (U_1, \ldots, U_{w_1})$ from the challenger of SMH-security game, it uses $U_{\tilde{\iota}}$ to compute the first component of the right hand side of the above equation i.e. $g_1^{\mathbf{B}\begin{pmatrix}\mathbf{0}\\c_{\tilde{\iota}}(\hat{\mathbf{s}},\hat{\mathbf{h}})\end{pmatrix}} = U_{\tilde{\iota}}^{\mathbf{B}\begin{pmatrix}\mathbf{0}\\1\end{pmatrix}}$.

However to compute the second component $\left(\prod_{\substack{j\in[0,w_2]\\k\in[n]}} g_1^{a_{\tilde{\iota}jk}\mathbf{H}_k\mathbf{B}\begin{pmatrix}\mathbf{0}\\\hat{s}_j\end{pmatrix}}\right)$ of the semi-functional part of the challenge ciphertext, $\mathcal{B}$ needs to know $\hat{\mathbf{s}} = (\hat{s}_0, \ldots, \hat{s}_{w_2})$. For each of $j \in [0, w_2]$, two cases can happen.

- Either there is $\tilde{\iota}' \in [w_1]$ such that $c_{\tilde{\iota}'}(\hat{\mathbf{s}}, \hat{\mathbf{h}}) = \hat{s}_j$, that lets $\mathcal{B}$ to know such an $\hat{s}_j$.
- Or there is no such $\tilde{\iota}' \in [w_1]$ for which $c_{\tilde{\iota}'}(\hat{\mathbf{s}}, \hat{\mathbf{h}}) = \hat{s}_j$. Then due to regular decryption properties of pair encoding (precisely Property $\mathcal{P}3$), $a_{\tilde{\iota}''jk} = 0$ for all $\tilde{\iota}'' \in [w_1]$, $k \in [n]$.

Due to regular decryption properties of pair encoding (precisely Property $\mathcal{P}4$), $\mathcal{B}$ also can compute the semi-functional component of the blinding factor $e(g_1^{\boldsymbol{\alpha}^\top\mathbf{B}\begin{pmatrix}\mathbf{0}\\\hat{s}_0\end{pmatrix}}, g_2)$ as $g_1^{\hat{s}_0}$ is available in $\mathbf{U}$.

$\mathcal{B}$ uses the normal part of the ciphertext and the semi-functional part of the ciphertext to generate the $\mathbf{C}^*$. Runs $(vk^*, sk^*) \leftarrow \mathsf{OTS.Gen}(1^\lambda)$. Then it evaluates $\xi^* = \mathcal{H}(\mathbf{C}^*, vk^*)$ to compute $\overline{\mathbf{C}}_0^*$ and defines $\overline{\mathbf{C}}^* = (\overline{\mathbf{C}}_0^*, \mathbf{C}^*, vk^*, \sigma^*)$ where $\sigma^* \leftarrow \mathsf{OTS.Sign}(sk^*, \overline{\mathbf{C}}_0)$. It outputs $\overline{\mathbf{C}}^*$.

Key Queries. On j^{th} secret key query x_j ($j \in [q_1 + 1, q]$) $\mathcal{B}$ forwards x_j as a key-query to the challenger to receive $\mathbf{V} = g_2^{\mathbf{k}(\beta,\hat{\mathbf{r}},\hat{\mathbf{h}})}$ where $(\mathbf{k} = (k_1, \ldots, k_{m_1}); m_2) \leftarrow \mathsf{EncK}(x_j, N)$. $\mathcal{B}$ has to decide if $\beta = 0$ or $\beta \xleftarrow{\$} \mathbb{Z}_p$. It is to be noted that $\hat{r}$ and $\hat{\mathbf{h}}$ are chosen by the challenger of SMH-security game, unknown to $\mathcal{B}$. So $\mathcal{B}$ computes the normal part of the key by computing $g_2^{k_\iota(\boldsymbol{\alpha},\mathbf{R},\mathbb{H})}$ for $\mathbf{R} = \left(\begin{pmatrix}\mathbf{r}_1\\0\end{pmatrix}, \ldots, \begin{pmatrix}\mathbf{r}_{m_2}\\0\end{pmatrix}\right)$ such that $\mathbf{r}_1, \ldots, \mathbf{r}_{m_2} \xleftarrow{\$} \mathbb{Z}_p^d$. To compute the semi-functional part, that contains $\hat{\mathbf{h}}$ which is unknown to $\mathcal{B}$, it implicitly sets $\hat{\mathbf{R}} = \left(\begin{pmatrix}\mathbf{0}\\\hat{r}_1\end{pmatrix}, \ldots, \begin{pmatrix}\mathbf{0}\\\hat{r}_{m_2}\end{pmatrix}\right)$ where $\hat{\mathbf{r}} = (\hat{r}_1, \ldots, \hat{r}_{m_2})$. Then it computes the semi-functional component of the key as following.

$$g_2^{\mathbf{k}'(\beta,\hat{\mathbf{R}},\mathbb{H},\hat{\mathbf{h}})} = \left\{ g_2^{\mathbf{Z}\begin{pmatrix}\mathbf{0}\\k_\iota(\beta,\hat{\mathbf{r}},\hat{\mathbf{h}})\end{pmatrix}} \prod_{\substack{j\in[m_2]\\k\in[n]}} g_2^{b_{\iota jk}\mathbf{H}_k^\top\mathbf{Z}\begin{pmatrix}\mathbf{0}\\\hat{r}_j\end{pmatrix}} \right\}_{\iota\in[m_1]}.$$

Since $\mathcal{B}$ already has received $\mathbf{V} = (V_1, \ldots, V_{m_1})$ from the challenger of SMH-security game, it uses V_ι to compute the first component of the right hand side of the above equation i.e. $g_2^{\mathbf{Z}\begin{pmatrix}\mathbf{0}\\k_\iota(\beta,\hat{\mathbf{r}},\hat{\mathbf{h}})\end{pmatrix}} = V_\iota^{\mathbf{Z}\begin{pmatrix}\mathbf{0}\\1\end{pmatrix}}$.

However to compute the second component $\left(\prod_{\substack{j\in[m_2]\\k\in[n]}} g_2^{b_{\iota jk}\mathbf{H}_k^\top\mathbf{Z}\begin{pmatrix}\mathbf{0}\\\hat{r}_j\end{pmatrix}}\right)$ of the semi-functional part of the secret key, $\mathcal{B}$ needs to know $\hat{\mathbf{r}} = (\hat{r}_1, \ldots, \hat{r}_{m_2})$. For each of $j \in [m_2]$, two cases can happen.

– Either there is $\iota' \in [m_1]$ such that $k_{\iota'}(\beta, \hat{\mathbf{r}}, \hat{\mathbf{h}}) = \hat{r}_j$, that lets $\mathcal{B}$ to know $\hat{r}_j$.
– Or there is no such $\iota' \in [m_1]$ for which $k_{\iota'}(\beta, \hat{\mathbf{r}}, \hat{\mathbf{h}}) = \hat{r}_j$. Then due to regular decryption properties of pair encoding (precisely Property $\mathcal{P}2$), $b_{\iota'' jk} = 0$ for all $\iota'' \in [m_1]$, $k \in [n]$.

$\mathcal{B}$ uses the normal part of the key and the semi-functional part of the key to generate the secret key and hands it over to $\mathcal{A}$.

Dec Queries. Same as Phase-I decryption queries.

Guess. $\mathcal{A}$ halts with output $\mathfrak{b}'$. $\mathcal{B}$ outputs 1 if $\mathfrak{b}' = \mathfrak{b}$ and 0 otherwise.

B.8 Type-2 to Type-3 Key in Phase-II

Lemma 11 ($\mathsf{Game}_{2,q_1+1,2}$ to $\mathsf{Game}_{2,q_1+1,3}$). *For any efficient adversary $\mathcal{A}$ that makes at most q_1 pre-challenge key queries, at most q_2 post-challenge key queries and at most $q_{_D}$ decryption queries, there exists a PPT algorithm $\mathcal{B}$ such that $|\mathsf{Adv}^{2,q_1+1,2}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{2,q_1+1,3}_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\mathcal{D}_d\text{-MatDH}}_{\mathcal{B}}(\lambda)$.*

Proof. The algorithm $\mathcal{B}$ gets as input $(\mathbf{G}, g_2^{\mathbf{T}}, g_2^{\mathbf{T}\begin{pmatrix}\boldsymbol{y}\\ \hat{y}\end{pmatrix}})$ as $\mathcal{D}_d$-MatDH problem instance where $\hat{y} = 0$ or $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$, $\boldsymbol{y} \xleftarrow{\$} \mathbb{Z}_p^d$.

The simulator description is same as Lemma 8 except the simulator has to generate all post-challenge keys at once. Here the simulator again uses random self-reducibility property of Matrix-DH problem to create $q_2 m_2$ many instance of the given problem. It uses first m_2 instances to answer $(q_1+1)^{th}$ key query, next m_2 instances to answer $(q_1+2)^{th}$ key query, and so on. For $\iota \in [m_2]$, each ι^{th} component of secret key of i^{th} key query is now multiplied by $g_2^{k_\iota(\mathbf{Z}\begin{pmatrix}\mathbf{0}\\ \beta\end{pmatrix}, \mathbf{0}, \mathbb{H})} \in \mathbb{G}_2^{(d+1)}$. As $\mathcal{B}$ knows $\widehat{mpk}_{\text{base}}$, it chooses only one $\beta \xleftarrow{\$} \mathbb{Z}_p$ to perform the simulation. Similar to the proof of Lemma 6, we see that if $\hat{\boldsymbol{y}} = \mathbf{0}$, the key is a type-3 key whereas if $\hat{\boldsymbol{y}} \xleftarrow{\$} \left(\mathbb{Z}_p\right)^{m_2}$, the key is type-2 key.

C An Alternative Construction

For a pair encoding scheme P for predicate function R, a predicate encryption Π'_R for predicate function R is defined as following.

– $\mathsf{Setup}(1^\lambda, N)$: mpk and msk is same as Section 4.2. Only difference is we no longer require OTS.
– $\mathsf{KeyGen}(msk, x)$: Same as KeyGen in Section 4.2.
– $\mathsf{Enc}(mpk, y, M)$: Same as Enc in Section 4.2. Only difference being the ciphertext it outputs is $\overline{\mathbf{C}} = (\overline{\mathbf{C}}_0, \mathbf{C})$ where $\mathbf{C}$ is computed exactly the same as presented in Enc in Section 4.2. However, in this construction, $\xi = \mathcal{H}(\mathbf{C})$ and $\overline{\mathbf{C}}_0 = g_1^{(\xi\mathbf{H}_{n+1} + \mathbf{H}_{n+2})\mathbf{B}\begin{pmatrix}\mathbf{s}_0\\ 0\end{pmatrix}}$.
– $\mathsf{Dec}(\mathbf{K}, \overline{\mathbf{C}})$: It differs from the Dec in Section 4.2. Given $\mathbf{K}$ and $\overline{\mathbf{C}}$ corresponding to key-index x and data-index y respectively, if $R(x, y) = 0$, it aborts. It then computes $\xi = \mathcal{H}(\mathbf{C})$. It aborts if Eq. (7) is not satisfied.

$$e(\overline{\mathbf{C}}_0, g_2^{\mathbf{Z}\begin{pmatrix}\mathbf{I}_d\\ \mathbf{0}\end{pmatrix}}) = e(\mathbf{C}_1, g_2^{(\xi\mathbf{H}_{n+1}^\top + \mathbf{H}_{n+2}^\top)\mathbf{Z}\begin{pmatrix}\mathbf{I}_d\\ \mathbf{0}\end{pmatrix}}). \tag{7}$$

Then runs $\mathbf{E} \leftarrow \mathsf{Pair}(x, y, N)$. Given $\mathbf{K} = (\mathbf{K}_1, \ldots, \mathbf{K}_{m_1})$ and ciphertext $\overline{\mathbf{C}}$ it computes $(\tilde{K}_1, \ldots, \tilde{K}_{w_1})$ where $\tilde{K}_{\tilde{\iota}} = \prod_{\iota \in [m_1]} (\mathbf{K}_\iota)^{\mathbf{E}_{\iota\tilde{\iota}}}$ for each $\tilde{\iota} \in [w_1]$. Chooses $\mathfrak{r} \xleftarrow{\$} \mathbb{Z}_p^d$. Defines modified key $\hat{\mathbf{K}} = (K_0, \Phi \cdot \tilde{K}_1, \tilde{K}_2, \ldots, \tilde{K}_{w_1})$ where $K_0 = g_2^{\mathbf{Z}\begin{pmatrix}\mathfrak{r}\\ 0\end{pmatrix}}$ and $\Phi = g_2^{(\xi\mathbf{H}_{n+1}^\top + \mathbf{H}_{n+2}^\top)\mathbf{Z}\begin{pmatrix}\mathfrak{r}\\ 0\end{pmatrix}}$ for $\xi = \mathcal{H}(\mathbf{C})$.

Outputs M such that

$$M = \mathbf{C}_{w_1+1} \cdot e(\overline{\mathbf{C}}_0, K_0) \cdot \left(\prod_{\tilde{\imath} \in [w_1]} e(\mathbf{C}_{\tilde{\imath}}, \hat{\mathbf{K}}_{\tilde{\imath}}) \right)^{-1}. \tag{8}$$

Correctness. This construction is correct due to the proof of correctness in Section 4.3 and associativity property of [Att16, Section 4.1] for the check performed in Eq. (7).

Efficiency. We introduce an extra check in Eq. (7) to ensure $\overline{\mathbf{C}}_0$ to have a particular structure. The check in Eq. (7) incurs additional $2 \times (d+1)$ pairing evaluations. Therefore this CPA-to-CCA conversion incurs $3 \times (d+1)$ pairing evaluations during decryption in addition to pairing evaluation involved in CPA-ciphertext decryption [Att16][3].

Analysis. Let $\overline{\mathbf{C}} = (\overline{\mathbf{C}}_0, \mathbf{C})$ be a ciphertext. We emphasize that if $\mathbf{C}_1 = g_1^{\mathbf{c}_1}$ where $\mathbf{c}_1 \in \mathbb{Z}_p^{(d+1)}$ and $\overline{\mathbf{C}}_0 = g_1^{(\xi \mathbf{H}_{n+1} + \mathbf{H}_{n+2})\mathbf{c}_1}$, then such a ciphertext $\overline{\mathbf{C}}$ will satisfy Eq. (7). However, this is not the only case that satisfies Eq. (7). Due to the relation $(\mathbf{I}_d \ \mathbf{0})\mathbf{Z}^\top \mathbf{B}\begin{pmatrix}\mathbf{0}\\1\end{pmatrix} = \mathbf{0}$, any $\overline{\mathbf{C}}' = (\overline{\mathbf{C}}'_0, \mathbf{C})$ where $\overline{\mathbf{C}}'_0 = g_1^{(\xi \mathbf{H}_{n+1} + \mathbf{H}_{n+2})\mathbf{c}_1 + \mathbf{B}\begin{pmatrix}0\\\tau\end{pmatrix}}$ for any $\tau \in \mathbb{Z}_p$ will also satisfy the Eq. (7). We discuss this in details below:

The RHS of Eq. (7) evaluates to $e(g_1, g_2)^{(\mathbf{I}_d \ \mathbf{0})\mathbf{Z}^\top (\xi \mathbf{H}_{n+1} + \mathbf{H}_{n+2})\mathbf{c}_1}$. A satisfied verification requires the LHS to evaluate the same. The exponent of the $\mathbb{G}_T$ element computed in Eq. (7) can be expressed as a system of linear equations $\mathbf{A}\mathbf{x} = \mathbf{V}$ where $\mathbf{A} = (\mathbf{I}_d \ \mathbf{0})\mathbf{Z}^\top \in \mathbb{Z}_p^{d \times (d+1)}$, $\mathbf{x} \in \mathbb{Z}_p^{(d+1)}$ and $\mathbf{V} = (\mathbf{I}_d \ \mathbf{0})\mathbf{Z}^\top (\xi \mathbf{H}_{n+1} + \mathbf{H}_{n+2})\mathbf{c}_1 \in \mathbb{Z}_p^d$. We can write $\mathbf{V} = \mathbf{A}\mathbf{x}'$ where $\mathbf{x}' = (\xi \mathbf{H}_{n+1} + \mathbf{H}_{n+2})\mathbf{c}_1$, it simply implies that $\mathbf{x}'$ is a solution of the system $\mathbf{A}\mathbf{x} = \mathbf{V}$.

Suppose there exists a system of linear equations $\mathbf{A}\mathbf{x} = \mathbf{V}$ where $\mathbf{A} \in \mathbb{Z}_p^{m \times n}$, $\mathbf{x} \in \mathbb{Z}_p^n$ and $\mathbf{V} \in \mathbb{Z}_p^m$ such that $\mathsf{Rank}(\mathbf{A}) = r \in \mathbb{N}$. We define the solution set of such linear system to be $S = \{\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{V}\}$ and the solution of corresponding homogeneous equations is $S_0 = \{\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{0}\}$. Naturally, if a solution $\mathbf{x}' \in S$ is available, then $S = \{\mathbf{x}' + \mathbf{x} : \mathbf{x} \in S_0\}$. Due to rank-nullity theorem, $n = \mathsf{Rank}(\mathbf{A}) + \mathsf{dim}(S_0)$. Therefore $\mathsf{dim}(S_0) = n - r$.

Here, in case of Eq. (7), we see that $r = \mathsf{Rank}(A) = d$ as $\mathbf{A} = (\mathbf{I}_d \ \mathbf{0})\mathbf{Z}^\top$ where $\mathbf{Z} \in \mathbb{Z}_p^{(d+1)\times(d+1)}$ is invertible and $n = (d+1)$. Therefore $\mathsf{dim}(S_0) = 1$. That means there exists non-trivial $\mathbf{x}_0 \in S_0$ and it spans the space S_0 alone. Now due to our construction, $(\mathbf{I}_d \ \mathbf{0})\mathbf{Z}^\top \mathbf{B}\begin{pmatrix}\mathbf{0}\\1\end{pmatrix} = \mathbf{0}$. Therefore $\mathbf{x}_0 = \mathbf{B}\begin{pmatrix}\mathbf{0}\\1\end{pmatrix}$ is a solution of homogeneous equation. As $\mathsf{dim}(S_0) = 1$, clearly $\{\mathbf{x}_0\}$ is the basis of S_0. Thus $S_0 = \left\{\mathbf{B}\begin{pmatrix}\mathbf{0}\\\tau\end{pmatrix} : \tau \in \mathbb{Z}_p\right\}$. Therefore $S = \left\{(\xi \mathbf{H}_{n+1} + \mathbf{H}_{n+2})\mathbf{c}_1 + \mathbf{B}\begin{pmatrix}\mathbf{0}\\\tau\end{pmatrix} : \tau \in \mathbb{Z}_p\right\}$.

Intuitively, the collision resistance of $\mathcal{H}$ does not allow the adversary to come up with a different $\mathbf{C}$ that results in the same *commitment* ξ. The adversary, after receiving challenge $\overline{\mathbf{C}}^* = (\overline{\mathbf{C}}^*_0, \mathbf{C}^*)$, can however keep the same $\mathbf{C}^*$ and construct $\overline{\mathbf{C}}'_0 = \overline{\mathbf{C}}_0 \cdot g_1^{\mathbf{B}\begin{pmatrix}\mathbf{0}\\\tau\end{pmatrix}}$ (for some $\tau \in \mathbb{Z}_p$) and produce $\overline{\mathbf{C}} = (\overline{\mathbf{C}}'_0, \mathbf{C}^*)$ as decryption query. Such a scenario allows the simulator to solve the $\mathcal{D}_d$-MatDH problem. Therefore during the security game, what the adversary can do is to come up with random ciphertext $\overline{\mathbf{C}}$ for decryption. With all but negligible probability, x used in decryption query $(x, \overline{\mathbf{C}})$ will not satisfy y which is implicit data-index of $\overline{\mathbf{C}}$. This way we are ultimately stopping the adversary to gather any non-trivial information.

C.1 Security Argument

Here we give hybrid security argument to prove the security of predicate encryption scheme Π'_R. We follow the same sequence of games described in Section 5. We note that the games are quite similar to the game descriptions in Section 5 where only difference is here we no longer require one-time signature and the ciphertext now is $\overline{\mathbf{C}} = (\overline{\mathbf{C}}_0, \mathbf{C})$. Here we present $\mathsf{Game}_{3,i,1}$, $\mathsf{Game}_{3,i,2}$, $\mathsf{Game}_{3,i,3}$ for $1 \le i \le q_{_D}$ of Table 1 as rest of the games will be similar to Appendix B. As described in Section 5, in $\mathsf{Game}_{3,i,j}$, the i^{th} altKey is type-j semi-functional where $i \in [q_{_D}]$ and $j \in \{1,2,3\}$.

[3] The construction in Section 4.2 needs only $(d+1)$ additional pairing evaluations and a signature verification.

Note that in all of these above mentioned games, the decryption query can only be made on ciphertext $\overline{\mathbf{C}}$ where $\overline{\mathbf{C}}_0 = g_1^{(\xi \mathbf{H}_{n+1} + \mathbf{H}_{n+2})\mathbf{c}_1}$ and $\mathbf{C}_1 = g_1^{\mathbf{c}_1}$. The reason is discussed in Footnote 4 in Lemma 12.

Theorem 2. *Suppose a regular decryption pair encoding scheme P for predicate R is both* SMH *and* CMH*-secure in $\mathcal{G}$, and the $\mathcal{D}_d$-Matrix DH Assumption holds in $\mathcal{G}$. Then the scheme Π'_R (in Appendix C) is fully CCA-secure encryption scheme if $\mathcal{H}$ is collision resistant hash function. More precisely, for any PPT adversary $\mathcal{A}$ that makes at most q_1 key queries before challenge, at most q_2 key queries after challenge and at most $q_{_D}$ decryption queries throughout the game, there exists PPT algorithms $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4$ such that for any λ,*

$$\mathsf{Adv}^{\mathrm{PE}}_{\mathcal{A}}(\lambda) \leq (2q_1 + 2q_{_D} + 3) \cdot \mathsf{Adv}^{\mathcal{D}_d\text{-MatDH}}_{\mathcal{B}_1}(\lambda) + q_1 \cdot \mathsf{Adv}^{\mathrm{CMH}}_{\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathrm{SMH}}_{\mathcal{B}_3}(\lambda) + q_{_D} \cdot \mathsf{Adv}^{\mathrm{CRH}}_{\mathcal{B}_4}(\lambda).$$

C.2 Normal to Type-1 altKey

Lemma 12 ($\mathsf{Game}_{3,i-1,3}$ to $\mathsf{Game}_{3,i,1}$). *For $i = 1, \ldots, q_{_D}$, for any efficient adversary $\mathcal{A}$ that makes at most q key queries and at most $q_{_D}$ decryption queries, there exists a PPT algorithm $\mathcal{B}$ such that $|\mathsf{Adv}^{3,i-1,3}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{3,i,1}_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\mathcal{D}_d\text{-MatDH}}_{\mathcal{B}}(\lambda)$.*

Proof. The algorithm $\mathcal{B}$ gets input $(\mathbf{G}, g_2^{\mathbf{T}}, g_2^{\mathbf{T}\begin{pmatrix}\boldsymbol{y}\\ \hat{y}\end{pmatrix}})$ as $\mathcal{D}_d$-MatDH problem instance where $\hat{y} = 0$ or $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$, $\boldsymbol{y} \xleftarrow{\$} \mathbb{Z}_p^d$.

Setup. Same as Lemma 1. Only difference is we do not use any OTS.

Key Queries. Same as Lemma 1.

Dec Queries. On j^{th} decryption query $(x, \overline{\mathbf{C}})$ where $\overline{\mathbf{C}}$ is a ciphertext on data-index y, if $R(x, y) \neq 1$, aborts. Otherwise $\mathcal{B}$ computes altKey $\hat{\mathbf{K}}$ and returns $\mathsf{AltDec}(\overline{\mathbf{C}}, \hat{\mathbf{K}})$ to $\mathcal{A}$. We now describe the altKey generation procedure.

- If $j > i$, it is normal altKey. As $\mathcal{B}$ knows msk, it computes the altKey $\hat{\mathbf{K}} \leftarrow \mathsf{AltKeyGen}(\overline{\mathbf{C}}, x, msk)$.
- If $j < i$, it is type-3 semi-functional altKey. $\mathcal{B}$ computes type-3 altKey $\hat{\mathbf{K}} \leftarrow \mathsf{SFAltKeyGen}(\overline{\mathbf{C}}, x, msk, -, \widehat{mpk}_{base}, 3, \eta)$.
- If $j = i$, it runs $(\mathbf{k} = (k_1, \ldots, k_{m_1}); m_2) \leftarrow \mathsf{EncK}(x, N)$ and $\mathbf{E} \leftarrow \mathsf{Pair}(x, y, N)$. Chooses $\mathbf{r}_1, \ldots, \mathbf{r}_{m_2} \xleftarrow{\$} \mathbb{Z}_p^d$ and defines $\mathbf{R} = \left(\begin{pmatrix}\mathbf{r}_1\\ 0\end{pmatrix}, \ldots, \begin{pmatrix}\mathbf{r}_{m_2}\\ 0\end{pmatrix}\right)$. It generates normal key $\mathbf{K} = (\mathbf{K}_1, \ldots, \mathbf{K}_{m_1})$ where for each $\iota \in [m_1]$, $\mathbf{K}_\iota = g_2^{k_\iota(\boldsymbol{\alpha}, \mathbf{R}, \mathbb{H})} = g_2^{\left(b_\iota \boldsymbol{\alpha} + \sum\limits_{j \in [m_2]} b_{\iota j} \mathbf{Z}\begin{pmatrix}\mathbf{r}_j\\ 0\end{pmatrix} + \sum\limits_{\substack{j \in [m_2]\\ k \in [n]}} b_{\iota j k} \mathbf{H}_k^\top \mathbf{Z}\begin{pmatrix}\mathbf{r}_j\\ 0\end{pmatrix}\right)}$. It then computes $(\tilde{K}_1, \ldots, \tilde{K}_{w_1})$ where $\tilde{K}_{\tilde{\iota}} = \prod\limits_{\iota \in [m_1]} (\mathbf{K}_\iota)^{\mathbf{E}_{\iota\tilde{\iota}}}$ for each $\tilde{\iota} \in [w_1]$.

  Here we note that the decryption queries need to follow a certain structure given in the footnote[4].

  Given $\overline{\mathbf{C}} = (\overline{\mathbf{C}}_0, \mathbf{C})$, $\mathcal{B}$ computes $\xi = \mathcal{H}(\mathbf{C})$. To compute the altKey, it implicitly sets $\tilde{\mathbf{Z}}^{-1}\begin{pmatrix}\boldsymbol{y}\\ \hat{y}\end{pmatrix} = \begin{pmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{pmatrix}$. Therefore $g_2^{\mathbf{Z}\begin{pmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{pmatrix}} = g_2^{\tilde{\mathbf{B}}^{-\top}\mathbf{T}\begin{pmatrix}\boldsymbol{y}\\ \hat{y}\end{pmatrix}}$. The simulator then computes modified key $\hat{\mathbf{K}} = (K_0, \Phi \cdot \tilde{K}_1, \tilde{K}_2, \ldots, \tilde{K}_{w_1})$ where $K_0 = g_2^{\mathbf{Z}\begin{pmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{pmatrix}}$, $\Phi = g_2^{(\xi \mathbf{H}_{n+1}^\top + \mathbf{H}_{n+2}^\top)\mathbf{Z}\begin{pmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{pmatrix}}$ and therefore is efficiently computable. It is evident from the description that if $\hat{y} = 0$, the key is a normal altKey whereas if $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$, the key is type-1 altKey.

---

[4] Suppose the queried ciphertext is $\overline{\mathbf{C}} = (\overline{\mathbf{C}}_0, \mathbf{C})$ where $\overline{\mathbf{C}}_0 = g_1^{(\xi \mathbf{H}_{n+1} + \mathbf{H}_{n+2})\mathbf{c}_1 + \mathbf{B}\begin{pmatrix}\mathbf{0}\\ \tau\end{pmatrix}}$ for some $\tau \in \mathbb{Z}_p$ and $\mathbf{C}_1 = g_1^{\mathbf{c}_1}$. Note that it satisfies the verification in Eq. (7). However, as the simulator knows $\mathbf{H}_{n+1}$ and $\mathbf{H}_{n+2}$, it can compute $Q = g_1^{(\xi \mathbf{H}_{n+1} + \mathbf{H}_{n+2})\mathbf{c}_1}$. Therefore it gets hold of $g_1^{\mathbf{B}\begin{pmatrix}\mathbf{0}\\ \tau\end{pmatrix}}$ by computing $\overline{\mathbf{C}}_0/Q$. Since, $\mathbf{B}$ and $\mathbf{Z}$ are simulated exactly as Lemma 1 (see the **Setup** of Lemma 1), and $\mathcal{B}$ implicitly sets $\tilde{\mathbf{Z}}^{-1}\begin{pmatrix}\boldsymbol{y}\\ \hat{y}\end{pmatrix} = \begin{pmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{pmatrix}$ to compute i^{th} altKey, $e\left(g_1^{\mathbf{B}\begin{pmatrix}\mathbf{0}\\ \tau\end{pmatrix}}, g_2^{\mathbf{Z}\begin{pmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{pmatrix}}\right)$ evaluation will allow the simulator to decide the $\mathcal{D}_d$-MatDH problem instance. Thus, under $\mathcal{D}_d$-MatDH assumption, the adversary can't make such decryption query. Therefore any decryption query $\mathcal{A}$ makes, to satisfy Eq. (7), the queried ciphertext $\overline{\mathbf{C}}$ must follow the relation that $\overline{\mathbf{C}}_0 = g_1^{(\xi \mathbf{H}_{n+1} + \mathbf{H}_{n+2})\mathbf{c}_1}$ and $\overline{\mathbf{C}}_1 = g_1^{\mathbf{c}_1}$ where $\xi = \mathcal{H}(\mathbf{C})$.

Challenge. Same as Lemma 1. Since we do not use any OTS, ξ^* now is $\mathcal{H}(\mathbf{C}^*)$ and $\overline{\mathbf{C}}^* = (\overline{\mathbf{C}}_0^*, \mathbf{C}^*)$ where $\overline{\mathbf{C}}_0^* = g_1^{(\xi^*\mathbf{H}_{n+1}+\mathbf{H}_{n+2})\mathbf{B}\begin{pmatrix}\mathbf{s}_0\\ \hat{s}_0\end{pmatrix}}$.

Key Queries. Same as Phase-I key queries.

Dec Queries. Same as Phase-I dec queries.

Guess. $\mathcal{A}$ halts with output $\mathfrak{b}'$. $\mathcal{B}$ outputs 1 if $\mathfrak{b}' = \mathfrak{b}$ and 0 otherwise.

C.3 Type-1 to Type-2 altKey

Lemma 13 ($\mathsf{Game}_{3,i,1}$ to $\mathsf{Game}_{3,i,2}$). *For $i = 1, \ldots, d$, we have $|\mathsf{Adv}_{3,i,1}(\lambda) - \mathsf{Adv}_{3,i,2}(\lambda)| = 0$ if $\mathcal{H}$ is Collision Resistant Hash Function.*

To prove the indistinguishability of the two games, we use the modified $\mathsf{SFSetup}$ namely $\mathsf{SFSetup}'$ (See Appendix B.3) that was used to prove indistinguishability of Lemma 7 and Lemma 10. Intuitively, to argue the indistinguishability, we introduce new randomness using $\mathsf{SFSetup}'$. Note that this newly introduced randomness does not affect the public key mpk. Then we show that introduction of such new randomness allows us to argue the indistinguishability. Recall that the challenge ciphertext is semi-functional and is denoted by $\overline{\mathbf{C}}^*$, the secret keys $\mathbf{K}$ are all type-3 keys and the altKey, computed to answer i^{th} decryption query, is denoted by $\hat{\mathbf{K}}$.

Here we prove that joint distribution of $\{\mathbf{K}, \overline{\mathbf{C}}^*, \hat{\mathbf{K}}\}$ if $\hat{\mathbf{K}}$ is type-1 altKey is identical to joint distribution of $\{\mathbf{K}, \overline{\mathbf{C}}^*, \hat{\mathbf{K}}\}$ if $\hat{\mathbf{K}}$ is type-2 altKey. Note that $\overline{\mathbf{C}}^* = (\overline{\mathbf{C}}_0^*, \mathbf{C}^*)$ such that $\overline{\mathbf{C}}_0^* = g_1^{(\xi^*\mathbf{H}_{n+1}+\mathbf{H}_{n+2})\mathbf{B}\begin{pmatrix}\mathbf{s}_0\\ \hat{s}_0\end{pmatrix}+\mathbf{B}\begin{pmatrix}\mathbf{0}\\ (\xi^*\hat{h}_{n+1}+\hat{h}_{n+2})\hat{s}_0\end{pmatrix}}$ where $\xi^* = \mathcal{H}(\mathbf{C}^*)$. Now we prove our claim that, the joint distributions of $\{\mathbf{K}, \overline{\mathbf{C}}^*, \hat{\mathbf{K}}\}$ behaves identically for both type-1 and type-2 altKey $\hat{\mathbf{K}}$.

Claim. The joint distribution of $\{\mathbf{K}, \overline{\mathbf{C}}^*, \hat{\mathbf{K}}\}$ if $\hat{\mathbf{K}}$ is type-1 altKey is identical to joint distribution of $\{\mathbf{K}, \overline{\mathbf{C}}^*, \hat{\mathbf{K}}\}$ if $\hat{\mathbf{K}}$ is type-2 altKey.

Proof. Note that $\mathbf{K}$ is type-3 key in both the distributions and can be computed by the simulator as it knows msk and $\widehat{mpk}_{\text{base}}$. Due to linearity of pair encoding, the challenge ciphertext $\overline{\mathbf{C}}^*$ and the altKey $\hat{\mathbf{K}}$ can be expressed as product of normal component and semi-functional component. Since the simulator knows msk and can compute the normal components, it suffices to show that the joint distributions are identical if the joint distribution of semi-functional components of $\overline{\mathbf{C}}^*$ and $\hat{\mathbf{K}}$ are identically distributed.

Notice that due to the introduction of $\hat{\mathbf{h}}$ (See Appendix B.3), the semi-functional ciphertext component $\overline{\mathbf{C}}_0^{*\prime}$ and the term Φ' used in altKey, is affected. To prove our claim, it suffices to argue that the following two distributions $(\overline{\mathbf{C}}_0^{*\prime}, \Phi')$ are identically distributed:

$$\left\{ g_1^{(\xi^*\mathbf{H}_{n+1}+\mathbf{H}_{n+2})\mathbf{B}\begin{pmatrix}\mathbf{0}\\ \hat{s}_0\end{pmatrix}+\mathbf{B}\begin{pmatrix}\mathbf{0}\\ (\xi^*\hat{h}_{n+1}+\hat{h}_{n+2})\hat{s}_0\end{pmatrix}},\ g_2^{\mathbf{Z}\begin{pmatrix}\mathbf{0}\\ (\xi\hat{h}_{n+1}+\hat{h}_{n+2})\hat{\mathfrak{r}}\end{pmatrix}+(\xi\mathbf{H}_{n+1}^\top+\mathbf{H}_{n+2}^\top)\mathbf{Z}\begin{pmatrix}\mathbf{0}\\ \hat{\mathfrak{r}}\end{pmatrix}} \right\}.$$

$$\left\{ g_1^{(\xi^*\mathbf{H}_{n+1}+\mathbf{H}_{n+2})\mathbf{B}\begin{pmatrix}\mathbf{0}\\ \hat{s}_0\end{pmatrix}+\mathbf{B}\begin{pmatrix}\mathbf{0}\\ (\xi^*\hat{h}_{n+1}+\hat{h}_{n+2})\hat{s}_0\end{pmatrix}},\ g_2^{\mathbf{Z}\begin{pmatrix}\mathbf{0}\\ u\eta\end{pmatrix}+\mathbf{Z}\begin{pmatrix}\mathbf{0}\\ (\xi\hat{h}_{n+1}+\hat{h}_{n+2})\hat{\mathfrak{r}}\end{pmatrix}+(\xi\mathbf{H}_{n+1}^\top+\mathbf{H}_{n+2}^\top)\mathbf{Z}\begin{pmatrix}\mathbf{0}\\ \hat{\mathfrak{r}}\end{pmatrix}} \right\}.$$

By natural restriction $\overline{\mathbf{C}}^* \neq \overline{\mathbf{C}}$ where $\overline{\mathbf{C}}^*$ is challenge ciphertext and $\overline{\mathbf{C}}$ is ciphertext provided for decryption. Therefore $(\overline{\mathbf{C}}_0^*, \mathbf{C}^*) \neq (\overline{\mathbf{C}}_0, \mathbf{C})$.

Then any of the following two cases can happen,

1. $\overline{\mathbf{C}}_0^* \neq \overline{\mathbf{C}}_0$ and $\mathbf{C}^* = \mathbf{C}$: we show that such a case can't happen. Since $\mathbf{C}^* = \mathbf{C}$, $\xi^* = \xi$ and $\mathbf{C}_1 = \mathbf{C}_1^* = g_1^{\mathbf{c}_1^*}$ naturally. This implies $\overline{\mathbf{C}}_0 = g_1^{(\xi^*\mathbf{H}_{n+1}+\mathbf{H}_{n+2})\mathbf{c}_1^*} = \overline{\mathbf{C}}_0^*$ which is a contradiction.
2. $\mathbf{C}^* \neq \mathbf{C}$: the inequality $\mathbf{C}^* \neq \mathbf{C}$ implies $\xi^* \neq \xi$ (due to collision resistance of $\mathcal{H}$). Therefore $\xi^*\hat{h}_{n+1} + \hat{h}_{n+2}$ and $\xi\hat{h}_{n+1} + \hat{h}_{n+2}$ are pairwise independent as $\hat{h}_{n+1}$ and $\hat{h}_{n+2}$ are chosen uniformly at random. It implies that the semi-functional components of the ciphertext and altKey in $\text{Game}_{3,i,1}$ and $\text{Game}_{3,i,2}$ are identically distributed.

C.4 Type-2 to Type-3 altKey

Lemma 14 ($\mathsf{Game}_{3,i,2}$ **to** $\mathsf{Game}_{3,i,3}$)**.** *For $i = 1, \ldots, q_{_D}$, for any efficient adversary $\mathcal{A}$ that makes at most q key queries and at most $q_{_D}$ decryption queries, there exists a PPT algorithm $\mathcal{B}$ such that $|\mathsf{Adv}^{3,i-1,3}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{3,i,1}_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\mathcal{D}_d\text{-MatDH}}_{\mathcal{B}}(\lambda)$.*

Proof. The algorithm $\mathcal{B}$ gets input $(\mathbf{G}, g_2^{\mathbf{T}}, g_2^{\mathbf{T}\begin{pmatrix}\boldsymbol{y}\\ \hat{y}\end{pmatrix}})$ as $\mathcal{D}_{d}$-MatDH problem instance where $\hat{y} = 0$ or $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$, $\boldsymbol{y} \xleftarrow{\$} \mathbb{Z}_p^d$.

The simulator description is same as Lemma 6 except while answering i^{th} decryption query. Here the altKey component $\hat{\mathbf{K}}_1 = \Phi \cdot \tilde{K}_1$ where Φ is now multiplied by $g_2^{\mathbf{Z}\begin{pmatrix}\mathbf{0}\\ \eta u\end{pmatrix}} \in \mathbb{G}_2$ where $u = \sum_{\iota \in [m_1]} b_\iota \mathbf{E}_{\iota 1}$. As $\mathcal{B}$ knows $\widehat{mpk}_{\text{base}}$, it chooses $\eta \xleftarrow{\$} \mathbb{Z}_p$ to perform the simulation. In the similar light of Lemma 12, we see that if $\hat{y} = 0$, the altKey is a type-3 altKey whereas if $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$, it is type-2 altKey.