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

Sanjit Chatterjee, Sayantan Mukherjee, and Tapas Pandit

Department of Computer Science and Automation,
Indian Institute of Science, Bangalore
{sanjit,sayantanm,tapas}@iisc.ac.in

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) 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 large 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 first direct CCA-secure predicate encryption scheme requires exactly one additional ciphertext component and three additional units of pairing evaluation during decryption. The second construction requires exactly three additional ciphertext components but needs only one additional unit pairing evaluation during decryption. This is a significant improvement over conventional approach for CPA-to-CCA conversion in prime order groups.

1 Introduction

Predicate encryption (PE) is a new paradigm for public-key encryption that evaluates a predicate function $R : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ in the encrypted domain. Informally, in a PE system, a ciphertext $\mathbf{C}$ is associated with a data-index $y \in \mathcal{Y}$, a secret key $\mathbf{K}$ is associated with a key-index $x \in \mathcal{X}$ and the secret key $\mathbf{K}$ can decrypt the ciphertext $\mathbf{C}$ if and only if $R(x, y) = 1$. The simplest example is IBE [1] where R is an equality predicate.

Waters [2] introduced the dual system technique to construct adaptively secure predicate encryption schemes. Attrapadung [3] and Wee [4] 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* [3] and *predicate encoding* [4] 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 allowed the first-ever construction of PE for regular language, ABE with constant-size ciphertext etc. as presented in [3]. However, all these CPA-secure predicate encryptions were constructed in composite order groups.

Later Attrapadung [5] and Chen et al. [6] constructed adaptive CPA-secure predicate encryption schemes in the prime order groups using pair encoding and predicate encoding respectively. The construction [6] was even more modular due to the use of *dual system group* (DSG) [7]. Agrawal et al. [8, 9] integrated pair encoding and dual system group and introduced different security notions for pair encoding.

Motivation. All the aforementioned schemes aim at constructing CPA-secure predicate encryption. In various practical scenarios, however, CCA-security is assumed to be mandatory. One can use available generic techniques [10–13]

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. They therefore face problems of two-fold – (1) increased length of key-indices and data-indices which result in a *bigger* secret key and ciphertext due to *index transformation* [10, 12] and (2) extra cost to perform verifiability or delegation. We consider verifiability based approach as a benchmark since delegation is not known for most of the predicate encryptions. 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. Blömer et al. [14] proposed a direct CCA-secure predicate encryption from pair encodings in composite order groups. Their verifiability based check requires additional pairing operations which is nearly same as that required in underlying CPA-decryption. As an alternative, Nandi et al. [15] suggested a direct conversion to CCA-secure predicate encryptions from pair encodings. Even though that conversion is efficient and generic, it works 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. [16] and its descendants suggested certain frameworks to convert pairing based cryptosystems from composite order to prime order. However, to the best of our knowledge, such frameworks are not directly applicable for CPA to CCA conversion.

Our Contribution. In this work, we consider the pair encoding based CPA-secure predicate encryptions construction of [3, 5]. We propose two generic constructions of direct CCA-secure PE from pair encoding based CPA-secure PE. Both of our constructions are achieved in prime order group and neither uses the trick called *index transformation* [10, 12]. This results in more efficient CCA-secure PE in terms of the size of the ciphertext and number of pairing evaluations during decryption. Roughly speaking, given a CPA-secure predicate encryption, we create a hash of CPA-ciphertext and extend the idea of *injective encoding* [17] that adds a new ciphertext component as a "commitment" of CPA-ciphertext. We call this construction as *direct* CCA-secure construction as we do not conform to the traditional two-step approach of *index transformation* followed by *delegation* or *verifiability*.

Our first construction adds *only one* additional component to the ciphertext of [5] at the cost of *three* additional unit pairing evaluations during decryption. The second construction however is more efficient in terms of the number of pairing evaluations during decryption. The ciphertext in this construction adds *exactly three* additional components to the ciphertext of [5] namely a $(d+1)$-tuple made up of source group elements (i.e. an element of $\mathbb{G} = G_1^{(d+1)}$), an one-time signature (OTS) verification key and a signature. During decryption this construction needs *only one* additional unit of pairing evaluation along with an OTSverification. As we can see in Table 1, our generic techniques to construct CCA-secure predicate encryptions enjoy smaller (constant) increase in ciphertext size that naturally results in less number of pairing evaluations during decryption.

Technique	|Key|	|Ciphertext|	Decryption Cost	
delegation-based	$\mathcal{O}(\lvert x'\rvert)$	$\mathcal{O}(\lvert y_{vk}\rvert)$	$\mathscr{C}(\mathsf{Decrypt}(\mathbf{K}_{x_{vk}}, \mathbf{C}_{y_{vk}}))$	$\mathscr{C}(\mathsf{Delegate}(\mathbf{K}_{x'}, x', x_{vk}))$
verifiability-based			$\mathscr{C}(\mathsf{Decrypt}(\mathbf{K}_{x'}, \mathbf{C}_{y_{vk}}))$	$\mathscr{C}(\mathsf{Verify}(\mathbf{C}_{y_{vk}}, x', \epsilon_{vk}))$
Π_R (Section 3.2)	$\mathcal{O}(\lvert x\rvert)$	$\mathcal{O}(\lvert y\rvert)$	$\mathscr{C}(\mathsf{Decrypt}(\mathbf{K}_x, \mathbf{C}_y))$	3 unit pairing
Π'_R (Section 3.4)	$\mathcal{O}(\lvert x\rvert)$	$\mathcal{O}(\lvert y\rvert)$	$\mathscr{C}(\mathsf{Decrypt}(\mathbf{K}_x, \mathbf{C}_y))$	1 unit pairing

Table 1. Comparative study of efficiency.[1]

The idea of using OTS(resp. injective encoding) to achieve CCA-secure PKE/(H)-IBE was first introduced in [18] (resp. [17]). Our approach, while bearing some similarity, differ significantly from [18, 17] and their follow-up works in the context of Id-based encryption. This is because, the structure of pair encoding based general PE is much more involved than that of simpler equality predicate in IBE. The primary achievement of this paper over [15] is amalgamation of [17] (resp. [18]) with prime-order matrix-based construction of predicate encryption [5]. Our techniques

[1] For delegation/verifiability-based conversions, $x' \leftarrow \mathcal{T}_1(x), x_{vk} \leftarrow \mathcal{T}_2(x, vk), y_{vk} \leftarrow \mathcal{T}_3(y, vk)$ are transformed indices [13] and usually quite larger than input indices (x or y). Each ciphertext or key-component is a $(d+1)$-dimensional vector from $\mathbb{G} = G_1^{(d+1)}$ and $\mathbb{H} = G_2^{(d+1)}$ respectively. $\mathscr{C}(\cdot)$ denotes the cost function. The two-columns under "Decryption Cost" follows the convention that the *first* cell is underlying CPA-Decryption cost and *second* cell contains *additional* cost to achieve CCA-Decryption.

(Lemma 1, Lemma 2) demonstrate that simple primitives like [17, 18] can still be deployed to achieve CCA-security even when the ciphertext/key is of complicated matrix structure.

Organization of the Paper. Section 2 contains necessary definitions and notations that are followed in this paper. In Section 3 we describe two constructions to achieve CCA-secure predicate encryption. Section 4 concludes the paper. We also recollect few standard primitives and hard problem in Appendix A. Conventional verifiability-based CPA-to-CCA conversion is discussed in Appendix B to explain the cost of such conversion generically. The games in the security argument that we mimic (and update to achieve CCA-security) from [5] are presented in Appendix C. We defer the security proof of our second construction to Appendix D.

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 set S. 1^λ denotes the security parameter for $\lambda \in \mathbb{N}$. Any $\mathbf{x} \in S^{\ell\!\!\!/}$ is a $\ell\!\!\!/$-dimensional column vector. We use both $\mathbf{x} \in S^{1\times \ell\!\!\!/}$ and $\mathbf{x} \in \left(S\right)^{\ell\!\!\!/}$ to denote $\ell\!\!\!/$-dimensional row vectors. $\mathbb{GL}_{N,\ell}$ is the group of non-singular matrices with dimension $\ell \times \ell$ and the scalars are from $\mathbb{Z}_N$.

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 $x \in \mathcal{X}$.
- $\mathsf{Encrypt}(mpk, y, M)$ takes as input data-index $y \in \mathcal{Y}$ and a message $M \in \mathcal{M}$ and generates ciphertext $\mathbf{C}$.
- $\mathsf{Decrypt}(\mathbf{K}, \mathbf{C})$ 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 $\mathbf{C} \leftarrow \mathsf{Encrypt}(mpk, y, M)$, all $x \in \mathcal{X}$, all $\mathbf{K} \leftarrow \mathsf{KeyGen}(msk, x)$,

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

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.
- Phase-I Query: Queries are performed to available oracles as follows.
  - Key Query: Keygen oracle $\mathcal{O}_K$ returns $\mathbf{K} \leftarrow \mathsf{KeyGen}(msk, x)$ for a given key-index x.
  - Dec Query: Given $(x, \mathbf{C})$, decryption oracle $\mathcal{O}_D$ returns $\mathsf{Decrypt}(\mathbf{K}, \mathbf{C})$.
- 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 $\mathbf{C}^* \leftarrow \mathsf{Encrypt}(mpk, y^*, M_\mathfrak{b})$ for $\mathfrak{b} \xleftarrow{\$} \{0, 1\}$.
- Phase-II Query: Queries are performed to available oracles as follows.
  - Key Query: Given a key-index x such that $R(x, y^*) = 0$, keygen oracle $\mathcal{O}_K$ returns $\mathbf{K} \leftarrow \mathsf{KeyGen}(msk, x)$.
  - Dec Query: Given $(x, \mathbf{C})$, decryption oracle $\mathcal{O}_D$ returns $\mathsf{Decrypt}(\mathbf{K}, \mathbf{C})$ if the conditions $R(x, y^*) = 1$ and $\mathbf{C} = \mathbf{C}^*$ are not satisfied together.

– 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,

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

A predicate encryption scheme is said to be IND-CCA secure if for any efficient adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\Pi_R}(\lambda) \leq \mathsf{neg}(\lambda)$. If the decryption oracle is not available to the adversary, we call such security model as IND-CPA security model.

2.2 Pair Encoding Schemes

Attrapadung [3] introduced the notion of pair encoding scheme which was later [5] refined with the properties called *regularity* of pair encoding. Here we recall the definition of pair encoding [3] and will discuss regular properties of pair encoding in Section 3.1.

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:

– $\mathsf{Param}(\kappa) \to n$ which is number of *common variables* $\mathbf{w} = (\mathrm{w}_1, \ldots, \mathrm{w}_n)$ 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 local variables $\mathbf{r} = (r_1, \ldots, r_{m_2})$, common variables $\mathbf{w}$ and private variable α.

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

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

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

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 recall two natural properties [3] of the pair encoding scheme as follows.

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

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

2.2.1 Security Definitions

Perfect Security. Pair Encoding P is said to be *perfectly master-key hiding* (aka PMH-*Secure*) 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,

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

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

Computational Security. Two types of computational security notions CMH and SMH are defined in [5] 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.

$$\begin{aligned}\mathrm{Exp}_{\mathcal{G},\mathsf{G},\mathfrak{d},\alpha,t_1,t_2}(\lambda) : (G_1, G_2, G_\mathrm{T}, e, N) \leftarrow \mathcal{G}(\lambda);\ (g_1, g_2) \xleftarrow{\$} G_1 \times G_2,\\ \alpha \xleftarrow{\$} \mathbb{Z}_N, n \leftarrow \mathsf{Param}(\kappa), \mathbf{w} \xleftarrow{\$} \mathbb{Z}_N^n;\\ \mathrm{st} \leftarrow \mathcal{A}_1^{\mathcal{O}^1_{\mathsf{G},\mathfrak{d},\alpha,\mathbf{w}}(\cdot)}(g_1, g_2); \mathfrak{d}' \leftarrow \mathcal{A}_2^{\mathcal{O}^2_{\mathsf{G},\mathfrak{d},\alpha,\mathbf{w}}(\cdot)}(\mathrm{st})\end{aligned}$$

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

Adversary $\mathcal{A}$ has advantage $\mathsf{Adv}^{t_1,t_2,\mathsf{G},P}_{\mathcal{A}}(\lambda) = |\Pr[\mathrm{Exp}_{P,\mathcal{G},\mathsf{G},0,\alpha,t_1,t_2}(\lambda) = 1] - \Pr[\mathrm{Exp}_{P,\mathcal{G},\mathsf{G},1,\alpha,t_1,t_2}(\lambda) = 1]|$ against pair encoding P . 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). Similar to [5], we will use security notions like $(1,1)$-CMH and $(1, \mathsf{poly})$-SMH security to prove the construction secure.

3 CCA-secure Predicate Encryption from Pair Encoding

Here we present two *direct* constructions of CCA-secure predicate encryption scheme in prime-order groups from pair encoding scheme.

3.1 Regular Decryption Pair Encoding

As recalled earlier, the notion of *regular* properties of pair encoding was introduced in [5, Definition 1]. We also note that the decryption sufficiency property was discussed in [15, Conditions 3.1]. Here, we need the pair encoding to satisfy both these properties. We call a pair encoding that satisfies all the above mentioned properties, *regular decryption* pair encoding. The regular decryption properties of a pair encoding are listed below. The first four (precisely Properties $\mathcal{P}1, \mathcal{P}2, \mathcal{P}3, \mathcal{P}4$) denote the regular properties of pair encoding. Note that, these restrictions are quite natural and are observed in all the available pair encoding based predicate encryption constructions [3, 4, 6, 8, 5, 9].

The regular decryption properties of pair encoding are 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 $c_{\hat{\tilde{\iota}}} = s_j$.

($\mathcal{P}4$) : $c_1(\mathbf{s}, \mathbf{w}) = s_0$.
($\mathcal{P}5$) : 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 [5, 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 $\mathbf{K} \in (G_2^{(d+1)})^{m_1}$ and a pairing matrix $\mathbf{E} \in \mathbb{Z}_N^{m_1 \times w_1}$ (see Section 2.2 for description), the decryptor will compute an altKey $\hat{\mathbf{K}} = (\hat{\mathrm{K}}_0, \hat{\mathrm{K}}_1, \ldots, \hat{\mathrm{K}}_{w_1}) \in (G_2^{(d+1)})^{(w_1+1)}$. We restrict that α used in secret key ($\mathbf{K}$) generation affects only $\hat{\mathrm{K}}_1$ via Property $\mathcal{P}5$. We will be needing this property in the security argument.

Next we describe our first construction and prove its security. Later we describe another construction that achieves better efficiency during decryption. Both of the constructions are developed on top of [5] as described in Remark 2.

3.2 Construction Π_R: Smaller Ciphertext

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

- $\mathsf{Setup}(1^\lambda, N)$: Runs $(G_1, G_2, G_\mathrm{T}, e, p) \leftarrow \mathcal{G}(\lambda)$ where $\mathcal{G}$ is an asymmetric prime-order bilinear group generator. Picks $(g_1, g_2) \xleftarrow{\$} G_1 \times G_2$. Runs $n \leftarrow \mathsf{Param}(\kappa)$. Defines $\mathbb{W} = (\mathbf{W}_1, \ldots, \mathbf{W}_{n+2})$ where $\mathbf{W}_i \xleftarrow{\$} \mathbb{Z}_p^{(d+1)\times(d+1)}$ 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 $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p$. Keeps $msk = (g_2^{\boldsymbol{\alpha}}, \mathbf{B}, \mathbf{Z}, \mathbb{W})$ to be secret and computes,
$$mpk = \left(g_1^{\mathbf{B}\left(\begin{smallmatrix}\mathbf{I}_d\\ 0\end{smallmatrix}\right)}, g_1^{\mathbf{W}_1\mathbf{B}\left(\begin{smallmatrix}\mathbf{I}_d\\ 0\end{smallmatrix}\right)}, \ldots, g_1^{\mathbf{W}_{n+2}\mathbf{B}\left(\begin{smallmatrix}\mathbf{I}_d\\ 0\end{smallmatrix}\right)}, g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathbf{I}_d\\ 0\end{smallmatrix}\right)}, g_2^{\mathbf{W}_1^\top\mathbf{Z}\left(\begin{smallmatrix}\mathbf{I}_d\\ 0\end{smallmatrix}\right)}, \ldots, g_2^{\mathbf{W}_{n+2}^\top\mathbf{Z}\left(\begin{smallmatrix}\mathbf{I}_d\\ 0\end{smallmatrix}\right)}, e(g_1, g_2)^{\boldsymbol{\alpha}^\top\mathbf{B}\left(\begin{smallmatrix}\mathbf{I}_d\\ 0\end{smallmatrix}\right)}, \mathcal{H} \right).$$
- $\mathsf{KeyGen}(msk, x)$: 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 $\mathbf{K} = \{g_2^{k_\iota(\boldsymbol{\alpha}, \mathbf{R}, \mathbb{W})}\}_{\iota \in [m_1]} \in \left(G_2^{(d+1)}\right)^{m_1}$ where for each $\iota \in [m_1]$,
$$k_\iota(\boldsymbol{\alpha}, \mathbf{R}, \mathbb{W}) = b_\iota \boldsymbol{\alpha} + \sum_{j \in [m_2]} b_{\iota j} \mathbf{Z}\left(\begin{smallmatrix}\mathbf{r}_j\\ 0\end{smallmatrix}\right) + \sum_{\substack{j \in [m_2] \\ k \in [n]}} b_{\iota jk} \mathbf{W}_k^\top \mathbf{Z}\left(\begin{smallmatrix}\mathbf{r}_j\\ 0\end{smallmatrix}\right).$$
- $\mathsf{Encrypt}(mpk, y, M)$: 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 $\mathbf{C} = (\mathrm{C}_1, \ldots, \mathrm{C}_{w_1}, \mathrm{C}_{w_1+1})$ where for each $\tilde{\iota} \in [w_1]$, $\mathrm{C}_{\tilde{\iota}} = g_1^{c_{\tilde{\iota}}(\mathbf{S}, \mathbb{W})} \in G_1^{(d+1)}$ such that
$$c_{\tilde{\iota}}(\mathbf{S}, \mathbb{W}) = \sum_{j \in [0, w_2]} a_{\tilde{\iota} j} \mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_j\\ 0\end{smallmatrix}\right) + \sum_{\substack{j \in [0, w_2] \\ k \in [n]}} a_{\tilde{\iota} jk} \mathbf{W}_k \mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_j\\ 0\end{smallmatrix}\right) \text{ for } \tilde{\iota} \in [w_1]$$
and $\mathrm{C}_{w_1+1} = M \cdot e(g_1, g_2)^{\boldsymbol{\alpha}^\top \mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\ 0\end{smallmatrix}\right)}$. It outputs $\overline{\mathbf{C}} = (\overline{\mathrm{C}}_0, \mathbf{C})$ where $\xi = \mathcal{H}(\mathbf{C})$ and $\overline{\mathrm{C}}_0 = g_1^{(\xi \mathbf{W}_{n+1} + \mathbf{W}_{n+2})\mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\ 0\end{smallmatrix}\right)}$.
- $\mathsf{Decrypt}(\mathbf{K}, \overline{\mathbf{C}})$: Given $\mathbf{K}$ and $\overline{\mathbf{C}}$ corresponding to key-index x and data-index y respectively, if $R(x, y) = 0$, it aborts. It then computes $\xi = \mathcal{H}(\mathbf{C})$. It aborts if Eq. (1) is not satisfied.
$$e(\overline{\mathrm{C}}_0, g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathbf{I}_d\\ \mathbf{0}\end{smallmatrix}\right)}) = e(\mathrm{C}_1, g_2^{(\xi \mathbf{W}_{n+1}^\top + \mathbf{W}_{n+2}^\top)\mathbf{Z}\left(\begin{smallmatrix}\mathbf{I}_d\\ \mathbf{0}\end{smallmatrix}\right)}). \tag{1}$$
Then runs $\mathbf{E} \leftarrow \mathsf{Pair}(x, y, N)$. Given $\mathbf{K} = (\mathrm{K}_1, \ldots, \mathrm{K}_{m_1})$ and ciphertext $\overline{\mathbf{C}}$ it computes $(\tilde{\mathrm{K}}_1, \ldots, \tilde{\mathrm{K}}_{w_1})$ where $\tilde{\mathrm{K}}_{\tilde{\iota}} = \prod_{\iota \in [m_1]} (\mathrm{K}_\iota)^{\mathbf{E}_{\iota\tilde{\iota}}}$ for each $\tilde{\iota} \in [w_1]$. Chooses $\mathfrak{t} \xleftarrow{\$} \mathbb{Z}_p^d$. Defines modified key $\hat{\mathbf{K}} = (\hat{\mathrm{K}}_0, \hat{\mathrm{K}}_1, \ldots, \hat{\mathrm{K}}_{w_1})$ where $\hat{\mathrm{K}}_0 = g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{t}\\ 0\end{smallmatrix}\right)}$,

$\hat{\mathrm{K}}_1 = \Phi \cdot \tilde{\mathrm{K}}_1$ for $\Phi = g_2^{(\xi \mathbf{W}_{n+1}^\top + \mathbf{W}_{n+2}^\top)\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\0\end{smallmatrix}\right)}$ and $\xi = \mathcal{H}(\mathbf{C})$ and $\hat{\mathrm{K}}_i = \tilde{\mathrm{K}}_i$ for $i \in [2, w_1]$. Outputs M such that

$$M = \mathrm{C}_{w_1+1} \cdot e(\overline{\mathrm{C}}_0, \hat{\mathrm{K}}_0) \cdot \left(\prod_{\tilde{\iota} \in [w_1]} e(\mathrm{C}_{\tilde{\iota}}, \hat{\mathrm{K}}_{\tilde{\iota}}) \right)^{-1}. \tag{2}$$

3.2.1 Correctness The correctness of Π_R follows primarily from the ciphertext consistency check in Eq. (1) and *associativity* property [5]. We discuss that in this section in detail.

Consistency Check in Π_R. Recall in Eq. (1), a consistency check is performed. The decryption proceeds only if this check is successful. It is easy to see that a *correct* ciphertext will always satisfy the equation.

We now discuss the equation (Eq. (2) and Eq. (3)) to get back the message M. Intuitively, the correctness holds due to the *associativity* property [5]. Here we compute $e(\overline{\mathrm{C}}_0, \hat{\mathrm{K}}_0) \cdot \left(\prod_{\tilde{\iota} \in [w_1]} e(\mathrm{C}_{\tilde{\iota}}, \hat{\mathrm{K}}_{\tilde{\iota}}) \right)^{-1}$ to show the correctness of both the constructions.

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

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

$$\begin{aligned} &= e(\mathrm{C}_1, \hat{\mathrm{K}}_1) \cdot \prod_{\tilde{\iota} \in [2, w_1]} e(\mathrm{C}_{\tilde{\iota}}, \hat{\mathrm{K}}_{\tilde{\iota}}) \\ &= e\left(g_1^{\mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\0\end{smallmatrix}\right)}, \Phi \cdot \tilde{\mathrm{K}}_1 \right) \cdot \prod_{\tilde{\iota} \in [2, w_1]} e(\mathrm{C}_{\tilde{\iota}}, \hat{\mathrm{K}}_{\tilde{\iota}}) && \text{(due to Property } \mathcal{P}4 \text{ of regular decryption pair encoding)} \\ &= e\left(g_1^{\mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\0\end{smallmatrix}\right)}, \Phi \right) \cdot e\left(g_1^{\mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\0\end{smallmatrix}\right)}, \tilde{\mathrm{K}}_1 \right) \cdot \prod_{\tilde{\iota} \in [2, w_1]} e(\mathrm{C}_{\tilde{\iota}}, \tilde{\mathrm{K}}_{\tilde{\iota}}) \\ &= e\left(g_1^{\mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\0\end{smallmatrix}\right)}, g_2^{(\xi \mathbf{W}_{n+1}^\top + \mathbf{W}_{n+2}^\top)\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\0\end{smallmatrix}\right)} \right) \cdot e(\mathrm{C}_1, \tilde{\mathrm{K}}_1) \cdot \prod_{\tilde{\iota} \in [2, w_1]} e(\mathrm{C}_{\tilde{\iota}}, \tilde{\mathrm{K}}_{\tilde{\iota}}) \\ &= e\left(g_1^{\mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\0\end{smallmatrix}\right)}, g_2^{(\xi \mathbf{W}_{n+1}^\top + \mathbf{W}_{n+2}^\top)\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\0\end{smallmatrix}\right)} \right) \cdot \prod_{\tilde{\iota} \in [w_1]} e(\mathrm{C}_{\tilde{\iota}}, \tilde{\mathrm{K}}_{\tilde{\iota}}) \end{aligned}$$

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

$$\begin{aligned} &= e\left(g_1^{\mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\0\end{smallmatrix}\right)}, g_2^{(\xi \mathbf{W}_{n+1}^\top + \mathbf{W}_{n+2}^\top)\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\0\end{smallmatrix}\right)} \right) \cdot e\left(g_1^{\mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\0\end{smallmatrix}\right)}, g_2^{(\xi \mathbf{W}_{n+1}^\top + \mathbf{W}_{n+2}^\top)\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\0\end{smallmatrix}\right)} \right)^{-1} \\ &\quad \cdot \left(\prod_{\tilde{\iota} \in [w_1]} e(\mathrm{C}_{\tilde{\iota}}, \tilde{\mathrm{K}}_{\tilde{\iota}}) \right)^{-1} \\ &= \left(\prod_{\tilde{\iota} \in [w_1]} e(\mathrm{C}_{\tilde{\iota}}, \prod_{\iota \in [m_1]} (\mathrm{K}_\iota)^{\mathbf{E}_{\iota\tilde{\iota}}}) \right)^{-1} \\ &= \left(\prod_{\substack{\iota \in [m_1] \\ \tilde{\iota} \in [w_1]}} e(\mathrm{C}_{\tilde{\iota}}, \mathrm{K}_\iota)^{\mathbf{E}_{\iota\tilde{\iota}}} \right)^{-1} \end{aligned}$$

$= e(g_1, g_2)^{-\boldsymbol{\alpha}^\top \mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\0\end{smallmatrix}\right)}$ (due to correctness of [5])

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

Remark 1. Decrypt 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*. We define $\mathsf{AltKeyGen}(\overline{\mathbf{C}}, x, msk)$ to compute modified key $\hat{\mathbf{K}}$ and $\mathsf{AltDecrypt}(\overline{\mathbf{C}}, \hat{\mathbf{K}})$ computes RHS of Eq. (2). This essentially divides the functionality of Decrypt function as composition of these two functions and helps us to process decryption queries during the proof.

Remark 2. We have extended the CPA-secure predicate encryption construction of [5]. The common variables $\mathbb{W}$, in our construction, contain $n+2$ matrices whereas in [5] it contained n matrices. These extra two common variables $\mathbf{W}_{n+1}$ and $\mathbf{W}_{n+2}$ are used to compute a commitment of CPA-ciphertext $\mathbf{C}$. A hash of $\mathbf{C}$ is computed first and is binded to the randomness $\mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\0\end{smallmatrix}\right)$ using common variables $\mathbf{W}_{n+1}$ and $\mathbf{W}_{n+2}$. This results in an extra ciphertext component, namely, $\overline{\mathrm{C}}_0$. We then output $\overline{\mathbf{C}} = (\overline{\mathrm{C}}_0, \mathbf{C})$ as ciphertext. Notice that KeyGen algorithm is exactly the same as [5]. The Decrypt is modified to perform cancellation of the component $\overline{\mathrm{C}}_0$. To do that, we define altKey $\hat{\mathbf{K}}$ to contain $\hat{\mathrm{K}}_0$ and Φ. We use *associativity* [5, Section 4.1] to cancel the extra ciphertext component $\overline{\mathrm{C}}_0$ using $\hat{\mathrm{K}}_0$ and Φ. The cancellation is performed by introducing an extra unit of pairing evaluation $e(\overline{\mathrm{C}}_0, \hat{\mathrm{K}}_0)$ during decryption. Once such cancellation is performed, the decryption happens exactly like [5].

Efficiency. We introduce an extra check in Eq. (1) to ensure $\overline{\mathrm{C}}_0$ to have a particular structure. The check in Eq. (1) incurs additional $2 \times (d+1)$ pairing evaluations. Therefore our construction incurs $3 \times (d+1)$ pairing evaluations during decryption in addition to pairing evaluation involved in CPA-ciphertext decryption [5]. This is really efficient as opposed to traditional CPA to CCA conversions by [10, 11, 13] that need roughly two times $m_1 \times w_1 \times (d+1) \times (m_2+1) \times d$ many pairing evaluations. We have discussed exact cost of such conversions in Appendix B.

Remark 3. Our construction uses a structure similar to the injective encoding first introduced in [7] to achieve CCA-secure PKE/(H)IBE from CPA-secure (H)IBE. However, the application of such a structure is far from straight forward as the ciphertext consistency check in Eq. (1) above may result in false-positives due to the complicated matrix-based structures in the ciphertext. We deal with this issue in Lemma 1.

3.3 Security of Π_R

To prove our predicate encryption construction (Π_R) fully CCA-secure, we extend the proof technique of Attrapadung [5]. In 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.

Suppose that after receiving challenge ciphertext $\overline{\mathbf{C}}^* = (\overline{\mathrm{C}}_0^*, \mathbf{C}^*)$, adversary modifies it to $\overline{\mathbf{C}} = (\overline{\mathrm{C}}_0^{*\prime}, \mathbf{C}^*)$. Lemma 1 emphasizes that such a ciphertext $\overline{\mathbf{C}}$ can pass Equation (1) if and only if $\overline{\mathrm{C}}_0^{*\prime} = \overline{\mathrm{C}}_0^* \cdot g_1^{\mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\\tau\end{smallmatrix}\right)}$ (for some $\tau \in \mathbb{Z}_p$). If adversary comes up with such $\overline{\mathbf{C}}$, one can devise an efficient $\mathcal{D}_d$-MatDH solver (described in Footnote 3 in Lemma 2). Therefore we can assume that the adversary always query well-formed ciphertext to decrypt oracle. This fact plays a key role in Lemma 3.

Lemma 1. *Let $\overline{\mathbf{C}} = (\overline{\mathrm{C}}_0, \mathbf{C})$ be a ciphertext (possibly ill-formed). Then the ciphertext $\overline{\mathbf{C}}$ will satisfy Eq. (1) if and only if $\overline{\mathrm{C}}_0 = g_1^{(\xi \mathbf{W}_{n+1} + \mathbf{W}_{n+2}) c_1 + \mathbf{B}\left(\begin{smallmatrix}0\\\tau\end{smallmatrix}\right)}$ and $\mathrm{C}_1 = g_1^{c_1}$ for any $\tau \in \mathbb{Z}_p$.*

Proof. The sufficiency of this lemma follows from *associativity* and the relation $(\mathbf{I}_d \ \mathbf{0}) \mathbf{Z}^\top \mathbf{B}\left(\begin{smallmatrix}0\\1\end{smallmatrix}\right) = \mathbf{0}$.

The necessary part of this lemma is given as follows. The RHS of Eq. (1) evaluates to $e(g_1, g_2)^{(\mathbf{I}_d \ \mathbf{0}) \mathbf{Z}^\top (\xi \mathbf{W}_{n+1} + \mathbf{W}_{n+2}) c_1}$. A satisfied verification requires the LHS to evaluate the same. The exponent of the G_T element computed in Eq. (1)

can be expressed as a system of linear equations $\mathbf{Ax} = \mathbf{V}$ 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 $\mathbf{V} = (\mathbf{I}_d\ \mathbf{0})\mathbf{Z}^\top(\xi\mathbf{W}_{n+1} + \mathbf{W}_{n+2})c_1 \in \mathbb{Z}_p^d$. We can write $\mathbf{V} = \mathbf{Ax}'$ where $\mathbf{x}' = (\xi\mathbf{W}_{n+1} + \mathbf{W}_{n+2})c_1$, it simply implies that $\mathbf{x}'$ is a solution of the system $\mathbf{Ax} = \mathbf{V}$.

Suppose there exists a system of linear equations $\mathbf{Ax} = \mathbf{V}$ 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 $S = \{\mathbf{x} : \mathbf{Ax} = \mathbf{V}\}$ and the solution of corresponding homogeneous equations is $S_0 = \{\mathbf{x} : \mathbf{Ax} = \mathbf{0}\}$. 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. (1), we see that $r = \mathsf{Rank}(\mathbf{A}) = d$ 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, $(\mathbf{I}_d\ \mathbf{0})\mathbf{Z}^\top\mathbf{B}\binom{\mathbf{0}}{1} = \mathbf{0}$. Therefore $\mathbf{x}_0 = \mathbf{B}\binom{\mathbf{0}}{1}$ is a solution of homogeneous equation. As $\mathsf{dim}(S_0) = 1$, clearly $\{\mathbf{x}_0\}$ is the basis of S_0. Thus $S_0 = \left\{\mathbf{B}\binom{\mathbf{0}}{\tau} : \tau \in \mathbb{Z}_p\right\}$. Therefore $S = \left\{(\xi\mathbf{W}_{n+1} + \mathbf{W}_{n+2})c_1 + \mathbf{B}\binom{\mathbf{0}}{\tau} : \tau \in \mathbb{Z}_p\right\}$. □

Theorem 1. *Suppose a regular decryption pair encoding scheme P for predicate R is both* SMH*-Secure and* CMH*-Secure*[2] *in $\mathcal{G}$, and the $\mathcal{D}_d$-Matrix DH Assumption holds in $\mathcal{G}$. Then the scheme Π_R 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 decryption queries throughout the game, there exist PPT algorithms $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4$ such that for any λ,*

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

We give a hybrid argument to prove Theorem 1. 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 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). The indistinguishability of Game_0 and Game_4 is proven via the sequence of games of Table 2.

Games	Difference from Previous Game	Indistinguishability from Previous Game
Game_0	-	-
Game_1	challenge ciphertext is semi-functional	[5]
$\mathsf{Game}_{2,i-1,3}$	all the $(i-1)$ secret keys are type-3 key $(i \le q_1)$	[5]
$\mathsf{Game}_{2,i,1}$	i^{th} secret key is type-1 key $(i \le q_1)$	[5]
$\mathsf{Game}_{2,i,2}$	i^{th} secret key is type-2 key $(i \le q_1)$	[5]
$\mathsf{Game}_{2,i,3}$	i^{th} secret key is type-3 key $(i \le q_1)$	[5]
$\mathsf{Game}_{2,q_1+1,1}$	all post-challenge secret keys are type-1 key	[5]
$\mathsf{Game}_{2,q_1+1,2}$	all post-challenge secret keys are type-2 key	[5]
$\mathsf{Game}_{2,q_1+1,3}$	all post-challenge secret keys are type-3 key	[5]
$\mathsf{Game}_{3,i-1,3}$	all the $(i-1)$ altKeys are type-3 altKey $(i \le Q)$	Lemma 2
$\mathsf{Game}_{3,i,1}$	i^{th} altKey is type-1 altKey $(i \le Q)$	Lemma 2
$\mathsf{Game}_{3,i,2}$	i^{th} altKey is type-2 altKey $(i \le Q)$	Lemma 3
$\mathsf{Game}_{3,i,3}$	i^{th} altKey is type-3 altKey $(i \le Q)$	Lemma 4
Game_4	challenge ciphertext is semi-functional encryption of a random message	Lemma 5

Table 2. Outline of proof strategy

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 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 $i \in [Q]$ 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

[2] Here SMH means $(1, \mathsf{poly})$-SMH and CMH means $(1,1)$-CMH (see [3, 5]).

ciphertext is completely independent of $\mathfrak{b}$. 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 2, we have used the proof technique of [5] to argue indistinguishability of several games. However, the games that *deal with* changes in altKey and the final game are primary contributions of this work. We, however, have included the description of games that we have mimicked (and modified for our requirement) from [5] in the Appendix C. Here we concentrate only in $\mathsf{Game}_{3,i,1}$, $\mathsf{Game}_{3,i,2}$, $\mathsf{Game}_{3,i,3}$ for $i \in [Q]$ 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 point are normal. We then change the altKeys to type-3 semi-functional altKey one by one. For every $i \in [Q]$, 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.

Note that in all of these above mentioned games, the decryption query can only be made on ciphertext $\overline{\mathbf{C}}$ where $\overline{\mathrm{C}}_0 = g_1^{(\xi \mathbf{W}_{n+1} + \mathbf{W}_{n+2})c_1}$ and $\mathrm{C}_1 = g_1^{c_1}$. The reason is discussed in Footnote 3 in Lemma 2.

3.3.1 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 $\widehat{mpk}_{base} = g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\1\end{smallmatrix}\right)}$, $\widehat{mpk}_b = \left(e(g_1,g_2)^{\boldsymbol{\alpha}^\top \mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\1\end{smallmatrix}\right)}, g_1^{\mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\1\end{smallmatrix}\right)}, g_1^{\mathbf{W}_1\mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\1\end{smallmatrix}\right)}, \ldots, g_1^{\mathbf{W}_{n+2}\mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\1\end{smallmatrix}\right)}\right)$ and $\widehat{mpk}_z = \left(g_2^{\mathbf{W}_1^\top \mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\1\end{smallmatrix}\right)}, \ldots, g_2^{\mathbf{W}_{n+2}^\top \mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\1\end{smallmatrix}\right)}\right)$.
- $\mathsf{SFKeyGen}(x, msk, \widehat{mpk}_z, \widehat{mpk}_{base}, \mathrm{type}, \beta)$: 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 $\hat{r}_1, \ldots, \hat{r}_{m_2} \xleftarrow{\$} \mathbb{Z}_p$. 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
  $$\mathbf{K} = \begin{cases} g_2^{\mathbf{k}(\boldsymbol{\alpha},\mathbf{R},\mathbb{W}) + \mathbf{k}(\mathbf{0},\widehat{\mathbf{R}},\mathbb{W})} & \text{if type} = 1 \\ g_2^{\mathbf{k}(\boldsymbol{\alpha},\mathbf{R},\mathbb{W}) + \mathbf{k}(\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\\beta\end{smallmatrix}\right),\widehat{\mathbf{R}},\mathbb{W})} & \text{if type} = 2 \\ g_2^{\mathbf{k}(\boldsymbol{\alpha},\mathbf{R},\mathbb{W}) + \mathbf{k}(\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\\beta\end{smallmatrix}\right),\mathbf{0},\mathbb{W})} & \text{if type} = 3 \end{cases}$$
  where $\mathbf{k}(\boldsymbol{\alpha},\mathbf{R},\mathbb{W}) + \mathbf{k}(\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\\beta\end{smallmatrix}\right),\widehat{\mathbf{R}},\mathbb{W}) = \left\{ b_\iota \boldsymbol{\alpha} + b_\iota \mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\\beta\end{smallmatrix}\right) + \sum_{j\in[m_2]} b_{\iota j} \mathbf{Z}\left(\begin{smallmatrix}\mathbf{r}_j\\\hat{r}_j\end{smallmatrix}\right) + \sum_{\substack{j\in[m_2]\\k\in[n]}} b_{\iota jk} \mathbf{W}_k^\top \mathbf{Z}\left(\begin{smallmatrix}\mathbf{r}_j\\\hat{r}_j\end{smallmatrix}\right) \right\}_{\iota\in[m_1]}$.
- $\mathsf{SFEncrypt}(y, M, mpk, \widehat{mpk}_b)$: 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 $\hat{s}_0, \ldots, \hat{s}_{w_2} \xleftarrow{\$} \mathbb{Z}_p$. Then 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 $\mathbf{C} = (\mathrm{C}_1, \ldots, \mathrm{C}_{w_1}, \mathrm{C}_{w_1+1})$ where for $\tilde{\iota} \in [w_1]$,
  $$\mathrm{C}_{\tilde{\iota}} = g_1^{\mathbf{c}_{\tilde{\iota}}(\mathbf{S},\mathbb{W}) + \mathbf{c}_{\tilde{\iota}}(\widehat{\mathbf{S}},\mathbb{W})} = g_1^{\left(\sum_{j\in[0,w_2]} a_{\tilde{\iota}j} \mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_j\\\hat{s}_j\end{smallmatrix}\right) + \sum_{\substack{j\in[0,w_2]\\k\in[n]}} a_{\tilde{\iota}jk} \mathbf{W}_k \mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_j\\\hat{s}_j\end{smallmatrix}\right)\right)}$$
  and $\mathrm{C}_{w_1+1} = M \cdot e(g_1,g_2)^{\boldsymbol{\alpha}^\top \mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\\hat{s}_0\end{smallmatrix}\right)}$. Then it computes $\xi = \mathcal{H}(\mathbf{C})$ and outputs $\overline{\mathbf{C}} = (\overline{\mathrm{C}}_0, \mathbf{C})$.
- $\mathsf{SFAltKeyGen}(\overline{\mathbf{C}}, x, msk, \widehat{mpk}_z, \widehat{mpk}_{base}, \mathrm{type}, \eta)$: 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 $\mathbf{K} = \left\{g_2^{k_\iota(\boldsymbol{\alpha},\mathbf{R},\mathbb{W})}\right\}_{\iota\in[m_1]} \in \left(G_2^{(d+1)}\right)^{m_1}$ where $k_\iota(\boldsymbol{\alpha},\mathbf{R},\mathbb{W}) = b_\iota \boldsymbol{\alpha} + \sum_{j\in[m_2]} b_{\iota j} \mathbf{Z}\left(\begin{smallmatrix}\mathbf{r}_j\\0\end{smallmatrix}\right) + \sum_{\substack{j\in[m_2]\\k\in[n]}} b_{\iota jk} \mathbf{W}_k^\top \mathbf{Z}\left(\begin{smallmatrix}\mathbf{r}_j\\0\end{smallmatrix}\right)$ for $\iota \in [m_1]$. Then it computes $(\tilde{\mathrm{K}}_1, \ldots, \tilde{\mathrm{K}}_{w_1})$ where $\tilde{\mathrm{K}}_{\tilde{\iota}} = \prod_{\iota\in[m_1]} (\mathrm{K}_\iota)^{\mathbf{E}_{\iota\tilde{\iota}}}$ for $\tilde{\iota} \in [w_1]$.

Defines modified key $\hat{\mathbf{K}} = (\hat{\mathrm{K}}_0, \Phi \cdot \tilde{\mathrm{K}}_1, \tilde{\mathrm{K}}_2, \ldots, \tilde{\mathrm{K}}_{w_1})$ where

$$(\hat{\mathrm{K}}_0, \Phi) = \begin{cases} \left(g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{smallmatrix}\right)}, g_2^{(\xi\mathbf{W}_{n+1}^\top + \mathbf{W}_{n+2}^\top)\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{smallmatrix}\right)}\right) & \text{if type} = 1 \\ \left(g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{smallmatrix}\right)}, g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ \eta u\end{smallmatrix}\right)+(\xi\mathbf{W}_{n+1}^\top + \mathbf{W}_{n+2}^\top)\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{smallmatrix}\right)}\right) & \text{if type} = 2 \\ \left(g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\ 0\end{smallmatrix}\right)}, g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ \eta u\end{smallmatrix}\right)+(\xi\mathbf{W}_{n+1}^\top + \mathbf{W}_{n+2}^\top)\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\ 0\end{smallmatrix}\right)}\right) & \text{if type} = 3 \end{cases}$$

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

3.3.2 Sequence of Games Here we present indistinguishability of $\mathsf{Game}_{3,i,1}$, $\mathsf{Game}_{3,i,2}$, $\mathsf{Game}_{3,i,3}$ and Game_4 for $1 \le i \le Q$ of Table 2 in the following lemmas.

Lemma 2 ($\mathsf{Game}_{3,i-1,3}$ **to** $\mathsf{Game}_{3,i,1}$)**.** *For $i \in [Q]$, for any efficient adversary $\mathcal{A}$ that makes at most q key queries and at most Q decryption queries, there exists a PPT algorithm $\mathcal{B}_1$ such that* $|\mathsf{Adv}_{\mathcal{A}}^{3,i-1,3}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{3,i,1}(\lambda)| \le \mathsf{Adv}_{\mathcal{B}_1}^{\mathcal{D}_d\text{-MatDH}}(\lambda)$.

Proof. The algorithm $\mathcal{B}_1$ gets input $(\mathbf{G}, g_2^{\mathbf{T}}, g_2^{\mathbf{T}\left(\begin{smallmatrix}\mathbf{y}\\ \hat{y}\end{smallmatrix}\right)})$ as $\mathcal{D}_d$-MatDH problem instance where $\hat{y} = 0$ or $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$, $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_p^d$.

Setup. $\mathcal{B}_1$ chooses $\tilde{\mathbf{B}} \xleftarrow{\$} \mathbb{GL}_{p,d+1}, \mathbf{J} \xleftarrow{\$} \mathbb{GL}_{p,d}$ and sets

$$\mathbf{B} = \tilde{\mathbf{B}} \begin{pmatrix} \mathbf{I}_d & \mathbf{M}^{-\top}\mathbf{c}^\top \\ \mathbf{0} & -1 \end{pmatrix} \text{ and } \mathbf{D} = \begin{pmatrix} \mathbf{MJ} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} \text{ where } \mathbf{T} = \begin{pmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{c} & 1 \end{pmatrix} \text{ due to } \mathcal{D}_d\text{-MatDH assumption.}$$

Then it defines

$$\mathbf{Z} = \mathbf{B}^{-\top}\mathbf{D} = \tilde{\mathbf{B}}^{-\top} \begin{pmatrix} \mathbf{I}_d & \mathbf{0} \\ \mathbf{c}\mathbf{M}^{-1} & -1 \end{pmatrix} \begin{pmatrix} \mathbf{MJ} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} = \tilde{\mathbf{B}}^{-\top}\mathbf{T} \begin{pmatrix} \mathbf{J} & \mathbf{0} \\ \mathbf{0} & -1 \end{pmatrix}.$$

It then defines $\tilde{\mathbf{Z}} = \begin{pmatrix} \mathbf{J} & \mathbf{0} \\ \mathbf{0} & -1 \end{pmatrix}$ so that $\mathbf{Z} = \tilde{\mathbf{B}}^{-\top}\mathbf{T}\tilde{\mathbf{Z}}$. $\mathcal{B}_1$ 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 $\mathcal{B}_1$ chooses $\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)}$ and $\mathbb{W} = (\mathbf{W}_1, \ldots, \mathbf{W}_{n+2}) \xleftarrow{\$} \left(\mathbb{Z}_p^{(d+1)\times(d+1)}\right)^{(n+2)}$ and publishes public key mpk. Note that $\mathcal{B}_1$ 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 $R(x, y) \ne 1$, aborts. Otherwise $\mathcal{B}_1$ computes altKey $\hat{\mathbf{K}}$ and returns $\mathsf{AltDecrypt}(\overline{\mathbf{C}}, \hat{\mathbf{K}})$ to $\mathcal{A}$. Here we emphasize that the decryption queries will follow a certain structure given in the footnote[3]. We now describe the altKey generation procedure.

– If $j > i$, it is normal altKey. As $\mathcal{B}_1$ knows msk, it computes the altKey $\hat{\mathbf{K}} \leftarrow \mathsf{AltKeyGen}(\overline{\mathbf{C}}, x, msk)$.

[3] Suppose given ciphertext is $\overline{\mathbf{C}} = (\overline{\mathrm{C}}_0, \mathbf{C})$ where $\overline{\mathrm{C}}_0 = g_1^{(\xi\mathbf{W}_{n+1} + \mathbf{W}_{n+2})c_1 + \mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\ \tau\end{smallmatrix}\right)}$ for some $\tau \in \mathbb{Z}_p$ and $\mathrm{C}_1 = g_1^{c_1}$. Note that it satisfies the verification in Eq. (1) as can be seen in Lemma 1. However, as the simulator knows $\mathbf{W}_{n+1}$ and $\mathbf{W}_{n+2}$, it can compute $L = g_1^{(\xi\mathbf{W}_{n+1} + \mathbf{W}_{n+2})c_1}$. Therefore it gets hold of $g_1^{\mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\ \tau\end{smallmatrix}\right)}$ by computing $\overline{\mathrm{C}}_0/L$. Since, $\mathbf{B}$ and $\mathbf{Z}$ are simulated exactly as Lemma 2 (see the **Setup** of Lemma 2), and $\mathcal{B}_1$ implicitly sets $\tilde{\mathbf{Z}}^{-1}\left(\begin{smallmatrix}\mathbf{y}\\ \hat{y}\end{smallmatrix}\right) = \left(\begin{smallmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{smallmatrix}\right)$ to compute i^{th} altKey, $e\left(g_1^{\mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\ \tau\end{smallmatrix}\right)}, g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\ \hat{\mathfrak{r}}\end{smallmatrix}\right)}\right)$ 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. (1), the queried ciphertext $\overline{\mathbf{C}}$ must follow the relation that $\overline{\mathrm{C}}_0 = g_1^{(\xi\mathbf{W}_{n+1} + \mathbf{W}_{n+2})c_1}$ and $\overline{\mathrm{C}}_1 = g_1^{c_1}$ where $\xi = \mathcal{H}(\mathbf{C})$.

– If $j < i$, it is type-3 semi-functional altKey. $\mathcal{B}_1$ 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 $\mathbf{R} = \left(\binom{\mathbf{r}_1}{0}, \ldots, \binom{\mathbf{r}_{m_2}}{0}\right)$. It generates normal key $\mathbf{K} = (\mathrm{K}_1, \ldots, \mathrm{K}_{m_1})$ where for each $\iota \in [m_1]$,

$$\mathrm{K}_\iota = g_2^{k_\iota(\boldsymbol{\alpha}, \mathbf{R}, \mathbb{W})} = g_2^{\left(\begin{array}{c} b_\iota \boldsymbol{\alpha} + \sum\limits_{j \in [m_2]} b_{\iota j} \mathbf{Z}\binom{\mathbf{r}_j}{0} + \sum\limits_{\substack{j \in [m_2] \\ k \in [n]}} b_{\iota jk} \mathbf{W}_k^\top \mathbf{Z}\binom{\mathbf{r}_j}{0} \end{array}\right)}.$$

It then computes $(\tilde{\mathrm{K}}_1, \ldots, \tilde{\mathrm{K}}_{w_1})$ where $\tilde{\mathrm{K}}_{\tilde{\iota}} = \prod\limits_{\iota \in [m_1]} (\mathrm{K}_\iota)^{\mathbf{E}_{\iota\tilde{\iota}}}$ for each $\tilde{\iota} \in [w_1]$.

Given $\overline{\mathbf{C}} = (\overline{\mathrm{C}}_0, \mathbf{C})$, $\mathcal{B}_1$ computes $\xi = \mathcal{H}(\mathbf{C})$. To compute the altKey, it implicitly sets $\tilde{\mathbf{Z}}^{-1}\binom{\mathbf{y}}{\hat{y}} = \binom{\mathfrak{r}}{\hat{\mathfrak{r}}}$. Therefore $g_2^{\mathbf{Z}\binom{\mathfrak{r}}{\hat{\mathfrak{r}}}} = g_2^{\tilde{\mathbf{B}}^{-\top}\mathbf{T}\binom{\mathbf{y}}{\hat{y}}}$. The simulator then computes modified key $\hat{\mathrm{K}} = (\hat{\mathrm{K}}_0, \Phi \cdot \tilde{\mathrm{K}}_1, \tilde{\mathrm{K}}_2, \ldots, \tilde{\mathrm{K}}_{w_1})$ where $\hat{\mathrm{K}}_0 = g_2^{\mathbf{Z}\binom{\mathfrak{r}}{\hat{\mathfrak{r}}}}$, $\Phi = g_2^{(\xi\mathbf{W}_{n+1}^\top + \mathbf{W}_{n+2}^\top)\mathbf{Z}\binom{\mathfrak{r}}{\hat{\mathfrak{r}}}}$ 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), $\mathcal{B}_1$ 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 $\binom{\mathbf{s}'_j}{\hat{s}'_j} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)}$ and implicitly sets $\binom{\mathbf{s}_j}{\hat{s}_j} = \mathbf{B}^{-1}\binom{\mathbf{s}'_j}{\hat{s}'_j}$. Then $\mathcal{B}_1$ computes $\mathbf{C}^*$ as it knows $\boldsymbol{\alpha}$, $\mathbf{W}_1, \ldots, \mathbf{W}_{n+2}$. It evaluates $\xi^* = \mathcal{H}(\mathbf{C}^*)$ to compute $\overline{\mathbf{C}}^* = (\overline{\mathrm{C}}_0^*, \mathbf{C}^*)$ where $\overline{\mathrm{C}}_0^* = g_1^{(\xi^*\mathbf{W}_{n+1} + \mathbf{W}_{n+2})\binom{\mathbf{s}'_0}{\hat{s}'_0}}$ and outputs $\overline{\mathbf{C}}^*$.

Key Queries. Same as Phase-I key queries.

Dec Queries. Same as Phase-I dec queries.

Guess. $\mathcal{A}$ halts with output $\mathfrak{b}'$. $\mathcal{B}_1$ outputs 1 if $\mathfrak{b}' = \mathfrak{b}$ and 0 otherwise. □

Lemma 3 ($\mathsf{Game}_{3,i,1}$ **to** $\mathsf{Game}_{3,i,2}$)**.** *For $i \in [Q]$, for all adversary $\mathcal{A}$ we have $|\mathsf{Adv}_{\mathcal{A}}^{3,i,1}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{3,i,2}(\lambda)| = 0$ if $\mathcal{H}$ is Collision Resistant Hash Function.*

To prove the indistinguishability of the two games, we use modified $\mathsf{SFSetup}$ namely $\mathsf{SFSetup}'$ that introduces independent randomness in $\widehat{mpk}_b$ and $\widehat{mpk}_z$ by means of security property *parameter-hiding* [5, Lemma 2] which is undetectable to the adversary $\mathcal{A}$ (see Appendix C.3 for more details). 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 $\hat{\mathbf{K}}$. To prove the lemma, note that, it is sufficient to argue that the joint distribution of semi-functional ciphertext, semi-functional secret keys and semi-functional altKeys stays identical, independent of if the altKey is type-1 altKey or a type-2 altKey.

Precisely, 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 $\overline{\mathbf{C}}^* = (\overline{\mathrm{C}}_0^*, \mathbf{C}^*)$ such that $\overline{\mathrm{C}}_0^* = g_1^{(\xi^*\mathbf{W}_{n+1} + \mathbf{W}_{n+2})\mathbf{B}\binom{\mathbf{s}_0}{\hat{s}_0} + \mathbf{B}\binom{\mathbf{0}}{(\xi^*\hat{\mathbf{w}}_{n+1} + \hat{\mathbf{w}}_{n+2})\hat{s}_0}}$ 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 $\hat{\mathbf{w}}$ (see Appendix C.3), the semi-functional ciphertext component $\overline{\mathrm{C}}_0^{*\prime}$ and the term Φ' used in altKey, is affected. To prove our claim, it suffices to argue that the following two distributions $(\overline{\mathrm{C}}_0^{*\prime}, \Phi')$ are identically distributed:

$$\left\{ g_1^{(\xi^*\mathbf{W}_{n+1}+\mathbf{W}_{n+2})\mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\ \hat{s}_0\end{smallmatrix}\right)+\mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\ (\xi^*\hat{\mathrm{w}}_{n+1}+\hat{\mathrm{w}}_{n+2})\hat{s}_0\end{smallmatrix}\right)}, g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ (\xi\hat{\mathrm{w}}_{n+1}+\hat{\mathrm{w}}_{n+2})\hat{\mathfrak{r}}\end{smallmatrix}\right)+(\xi\mathbf{W}_{n+1}^\top+\mathbf{W}_{n+2}^\top)\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ \hat{\mathfrak{r}}\end{smallmatrix}\right)} \right\}$$

and

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

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 $(\overline{\mathrm{C}}_0^*, \mathbf{C}^*) \neq (\overline{\mathrm{C}}_0, \mathbf{C})$.

Then any of the following two cases can happen,

1. $\overline{\mathrm{C}}_0^* \neq \overline{\mathrm{C}}_0$ and $\mathbf{C}^* = \mathbf{C}$: we show that such a case can't happen. Since $\mathbf{C}^* = \mathbf{C}$, $\xi^* = \xi$ and $\mathrm{C}_1 = \mathrm{C}_1^* = g_1^{c_1^*}$ naturally. This implies $\overline{\mathrm{C}}_0 = g_1^{(\xi^*\mathbf{W}_{n+1}+\mathbf{W}_{n+2})c_1^*} = \overline{\mathrm{C}}_0^*$ 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 $\xi^*\hat{\mathrm{w}}_{n+1} + \hat{\mathrm{w}}_{n+2}$ and $\xi\hat{\mathrm{w}}_{n+1} + \hat{\mathrm{w}}_{n+2}$ are pairwise independent as $\hat{\mathrm{w}}_{n+1}$ and $\hat{\mathrm{w}}_{n+2}$ are chosen uniformly at random. It implies that the semi-functional components of the ciphertext and altKey in $\mathsf{Game}_{3,i,1}$ and $\mathsf{Game}_{3,i,2}$ are identically distributed. □

Lemma 4 ($\mathsf{Game}_{3,i,2}$ to $\mathsf{Game}_{3,i,3}$). *For $i \in [Q]$, for any efficient adversary $\mathcal{A}$ that makes at most q key queries and at most Q decryption queries, there exists a PPT algorithm $\mathcal{B}_1$ such that $|\mathsf{Adv}_{\mathcal{A}}^{3,i,2}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{3,i,3}(\lambda)| \leq \mathsf{Adv}_{\mathcal{B}_1}^{\mathcal{D}_d\text{-MatDH}}(\lambda)$.*

Proof. The algorithm $\mathcal{B}_1$ gets input $(\mathbf{G}, g_2^{\mathbf{T}}, g_2^{\mathbf{T}\left(\begin{smallmatrix}\mathbf{y}\\ \hat{y}\end{smallmatrix}\right)})$ as $\mathcal{D}_d$-MatDH problem instance where $\hat{y} = 0$ or $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$, $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_p^d$.

The simulator description is same as Lemma 2 except while answering i^{th} decryption query. Here the altKey component $\hat{\mathrm{K}}_1 = \Phi \cdot \tilde{\mathrm{K}}_1$ where Φ is now multiplied by $g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ \eta u\end{smallmatrix}\right)} \in \mathbb{H}$ where $u = \sum_{\iota \in [m_1]} b_\iota \mathbf{E}_{\iota 1}$. As $\mathcal{B}_1$ knows $\widehat{mpk}_{\text{base}}$, it chooses $\eta \xleftarrow{\$} \mathbb{Z}_p$ to perform the simulation. In the similar light of Lemma 2, 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. □

Lemma 5 ($\mathsf{Game}_{3,Q,3}$ to Game_4). *For any adversary $\mathcal{A}$, we have $|\mathsf{Adv}_{\mathcal{A}}^{3,Q,3}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{4}(\lambda)| = 0$.*

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. $\boldsymbol{\alpha} = \mathbf{Z}\left(\begin{smallmatrix}\boldsymbol{\delta}\\ \hat{\delta}\end{smallmatrix}\right)$ 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 $\hat{\mathrm{K}}_1$ 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, $u\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ \eta\end{smallmatrix}\right)$, 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 $\mathrm{C}_{w_1+1}^*$. The resultant $\mathrm{C}_{w_1+1}^*$ will be $M_{\mathfrak{b}} \cdot e(g_1, g_2)^{\boldsymbol{\alpha}^\top \mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\ \hat{s}_0\end{smallmatrix}\right)} = M_{\mathfrak{b}} \cdot e(g_1, g_2)^{(\boldsymbol{\delta}^\top \ \hat{\delta}+t)\mathbf{Z}^\top \mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\ \hat{s}_0\end{smallmatrix}\right)} = M_{\mathfrak{b}} \cdot e(g_1, g_2)^{\boldsymbol{\alpha}^\top \mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\ \hat{s}_0\end{smallmatrix}\right)} \cdot e(g_1, g_2)^{(\mathbf{0}\ t)\mathbf{Z}^\top \mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\ \hat{s}_0\end{smallmatrix}\right)} = M_{\mathfrak{b}} \cdot e(g_1, g_2)^{\boldsymbol{\alpha}^\top \mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\ \hat{s}_0\end{smallmatrix}\right)} \cdot e(g_1, g_2)^{t\hat{s}_0}$. Therefore $\mathrm{C}_{w_1+1}^*$ encrypts $M_{\mathfrak{b}} \cdot e(g_1, g_2)^{t\hat{s}_0}$ that is an uniformly random element of G_{T} as $t \xleftarrow{\$} \mathbb{Z}_p$. □

3.4 Construction Π'_R: More Efficient Decryption

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

– $\mathsf{Setup}(1^\lambda, N)$: mpk and msk are same as Π_R in Section 3.2. Only difference is mpk now includes a one-time signature scheme $\mathsf{OTS}$of its choice.

– $\mathsf{KeyGen}(msk, x)$: Same as KeyGen of Π_R in Section 3.2.

– $\mathsf{Encrypt}(mpk, y, M)$: Same as $\mathsf{Encrypt}$ of Π_R in Section 3.2. Only difference being it runs $(vk, sk) \leftarrow \mathsf{OTS.Gen}(1^\lambda)$. Then it computes $\xi = \mathcal{H}(\mathbf{C}, vk)$ where $\mathbf{C}$ is computed exactly the same as presented in $\mathsf{Encrypt}$ of Π_R in Section 3.2 and outputs $\overline{\mathbf{C}} = (\overline{\mathrm{C}}_0, \mathbf{C}, vk, \sigma)$ where $\overline{\mathrm{C}}_0 = g_1^{(\xi \mathbf{W}_{n+1} + \mathbf{W}_{n+2})\mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\0\end{smallmatrix}\right)}$ and $\sigma \leftarrow \mathsf{OTS.Sign}(sk, \overline{\mathrm{C}}_0)$.

– $\mathsf{Decrypt}(\mathbf{K}, \overline{\mathbf{C}})$: It differs from the $\mathsf{Decrypt}$ of Π_R in Section 3.2. Given $\mathbf{K}$ and $\overline{\mathbf{C}}$ corresponding to key-index x and data-index y respectively, if $R(x, y) = 0$, it aborts. Also aborts if $\mathsf{OTS.Verify}(\overline{\mathrm{C}}_0, vk, \sigma)$ evaluates to 0. Otherwise runs $\mathbf{E} \leftarrow \mathsf{Pair}(x, y, N)$. Given $\mathbf{K} = (\mathrm{K}_1, \ldots, \mathrm{K}_{m_1})$ and ciphertext $\overline{\mathbf{C}}$ it computes $(\tilde{\mathrm{K}}_1, \ldots, \tilde{\mathrm{K}}_{w_1})$ where $\tilde{\mathrm{K}}_{\tilde{\iota}} = \prod_{\iota \in [m_1]} (\mathrm{K}_\iota)^{\mathbf{E}_{\iota\tilde{\iota}}}$ for each $\tilde{\iota} \in [w_1]$. Chooses $\mathfrak{t} \xleftarrow{\$} \mathbb{Z}_p^d$. Defines modified key $\hat{\mathbf{K}} = (\hat{\mathrm{K}}_0, \hat{\mathrm{K}}_1, \ldots, \hat{\mathrm{K}}_{w_1})$ where $\hat{\mathrm{K}}_0 = g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{t}\\0\end{smallmatrix}\right)}$, $\hat{\mathrm{K}}_1 = \Phi \cdot \tilde{\mathrm{K}}_1$ for $\Phi = g_2^{(\xi \mathbf{W}_{n+1}^\top + \mathbf{W}_{n+2}^\top)\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{t}\\0\end{smallmatrix}\right)}$ and $\xi = \mathcal{H}(\mathbf{C}, vk)$ and $\hat{\mathrm{K}}_i = \tilde{\mathrm{K}}_i$ for $i \in [2, w_1]$. Outputs M such that

$$M = \mathrm{C}_{w_1+1} \cdot e(\overline{\mathrm{C}}_0, \hat{\mathrm{K}}_0) \cdot \left(\prod_{\tilde{\iota} \in [w_1]} e(\mathrm{C}_{\tilde{\iota}}, \hat{\mathrm{K}}_{\tilde{\iota}}) \right)^{-1}. \tag{3}$$

3.4.1 Correctness The decryption of Π'_R also computes $\mathrm{C}_{w_1+1} \cdot e(\overline{\mathrm{C}}_0, \hat{\mathrm{K}}_0) \cdot \left(\prod_{\tilde{\iota} \in [w_1]} e(\mathrm{C}_{\tilde{\iota}}, \hat{\mathrm{K}}_{\tilde{\iota}}) \right)^{-1}$ to get back message M. However, the decryption in Π'_R differs from Π_R by the technique used to check consistency of ciphertext. Therefore it is sufficient to discuss that correctness of the consistency check in Π'_R.

Consistency Check in Π'_R. However, in case of Π'_R the consistency check is performed via the Verify of OTS. A *correct* ciphertext will always pass the OTSverification.

Remark 4. This construction is quite similar to the one presented in Section 3.2. However, the ciphertext now has extra two elements. Here, we compute the hash of $(\mathbf{C}, vk)$ first and bind it to the randomness $\mathbf{B}\left(\begin{smallmatrix}\mathbf{s}_0\\0\end{smallmatrix}\right)$ using common variables $\mathbf{W}_{n+1}$ and $\mathbf{W}_{n+2}$ where vk is verification key for OTS. This results in an extra ciphertext component namely $\overline{\mathrm{C}}_0$. We then use the one-time signature $\mathsf{OTS}$to compute a signature σ on $\overline{\mathrm{C}}_0$ and output $\overline{\mathbf{C}} = (\overline{\mathrm{C}}_0, \mathbf{C}, vk, \sigma)$. Use of OTSensures *integrity* of $\overline{\mathrm{C}}_0$ thereby allowing us to get rid of extra verification step (precisely Eq. (1) in Section 3.2) that is needed to check the structure of $\overline{\mathrm{C}}_0$.

Remark 5. Even if our construction uses $\mathsf{OTS}$for CCA-security, the technique is quite different than that of CHK [18] and its descendants. All those schemes essentially depends on the key-delegation capability of the underlying CPA-secure (H)IBE. Yamada et al. [10] formalized this notion as property of *delegatability*. We note that not all pair encoding based predicate encryptions achieve this property (see Table 3). Even for schemes that achieve delegatability, one needs to apply *index transformers* [12] on both key and data index. This often makes the resultant ciphertext and secret key significantly large (see [12, Table 1] for details) thereby degrading the decryption performance.

Efficiency. Our construction increases the ciphertext length by exactly three components namely $\overline{\mathrm{C}}_0$, vk and σ will be returned along with the CPA-ciphertext $\mathbf{C}$ where $\overline{\mathrm{C}}_0 \in G_1^{(d+1)}$, vk is verification key of $\mathsf{OTS}$and σ is the signature for $\overline{\mathrm{C}}_0$ with respect to the signing key sk corresponding to vk. As we mentioned earlier, we have reused KeyGen of [5], therefore the secret key does not change. However $\mathsf{Decrypt}$ has to verify the signature σ and evaluate only one additional unit of pairing (namely $e(\overline{\mathrm{C}}_0, \hat{\mathrm{K}}_0)$). As both $\overline{\mathrm{C}}_0$ and $\hat{\mathrm{K}}_0$ are group elements having $(d+1)$-components, the decryption in our construction incurs an additional cost of $(d+1)$ pairing evaluations only. This is more efficient than Π_R (of Section 3.2).

3.5 Security of Π'_R

Theorem 2. *Suppose a regular decryption pair encoding scheme P for predicate R is both* SMH*-Secure and* CMH*-Secure in $\mathcal{G}$, and the $\mathcal{D}_d$-Matrix DH Assumption holds in $\mathcal{G}$. Then the scheme Π'_R 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 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,*

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

The theorem is formally proved in Appendix D.

3.6 Performance Comparison

In this section we provide concrete comparison, outlined in Table 1, of performance between *conventional* conversions [10–13] and our constructions over few examples. The table below compares *delegation*-based, *verifiability*-based and our (*direct*) CCA-construction technique on pair encoding based prime-order instantiation of CP-ABE, KP-ABE [19] and KP-FE for DFA [5].

Scheme	|Key|	|Ciphertext|	Decryption Cost	
CP-ABE [19]	$2(\|A\|+2\ell)+2$	$2(m+\ell)+1$	$(I+\ell)[\mathsf{E}]+(I+\ell+2)[\mathsf{P}]$	$(\|A\|+\ell+2)[\mathsf{E}]$
	$2\|A\|+2$	$2(m+\ell)+1$	$I[\mathsf{E}]+(I+2)[\mathsf{P}]$	$(2I+2\ell+1)d[\mathsf{P}]$
	$2\|A\|+2$	$2m+1$	$I[\mathsf{E}]+(I+2)[\mathsf{P}]$	$2[\mathsf{E}]+3[\mathsf{P}]$ $2[\mathsf{E}]+[\mathsf{P}]$
KP-ABE [19]	NA	NA	NA	NA
	$2m$	$\|A\|+\ell+1$	$I[\mathsf{E}]+(I+1)[\mathsf{P}]$	$(2I+2\ell)d[\mathsf{P}]$
	$2m$	$\|A\|+1$	$I[\mathsf{E}]+(I+1)[\mathsf{P}]$	$2[\mathsf{E}]+3[\mathsf{P}]$ $2[\mathsf{E}]+[\mathsf{P}]$
KP-FE for DFA[5]	$3(\|\mathsf{Tr}\|+\ell)+5$	$2(\|\omega\|+\ell)+5$	$(3(\|\omega\|+\ell)+5)[\mathsf{P}]$	$\mathcal{O}((\|\mathsf{Tr}\|+\|Q\|+\ell)(\|\mathsf{Tr}\|+d+\ell))[\mathsf{E}]$ $2(\|\omega\|+\ell)d[\mathsf{E}]+(3(\|\omega\|+\ell)+7)[\mathsf{P}]$
	$3\|\mathsf{Tr}\|+5$	$2\|\omega\|+5$	$(5+3\|\omega\|)[\mathsf{P}]$	$2[\mathsf{E}]+3[\mathsf{P}]$ $2[\mathsf{E}]+[\mathsf{P}]$

Table 3. Concrete comparison of efficiency.[4]

For all the candidate schemes considered in the table, $vk \in \{0,1\}^\ell$. For CP-ABE and KP-ABE, A denotes attribute set, Γ denotes the access structure such that $I \subset A$ are the attributes that satisfy Γ and *dummy attribute* set W to accommodate vk such that $|W| = 2\ell$, where Γ can be expressed as an LSSS matrix [19] of dimension $m \times k$. In case of KP-FE for DFA, $M = (Q, \Sigma, \mathsf{Tr}, q_0, F)$ denotes the DFA and ω denotes the string. Here $[\mathsf{E}]$ and $[\mathsf{P}]$ denote number of *unit* group multiplication and number of *unit* pairing evaluations respectively. By *unit* group multiplication (resp. pairing evaluation) we mean $d+1$ many group multiplication (resp. pairing evaluations) in a prime-order system of $d+1$ dimension. For standard assumptions like SXDH and $\mathsf{D\text{-}Linear}$, d is 1 and 2 respectively.

Note that in Table 3, we considered both KP-ABE and CP-ABE to be *small-universe*. The additional cost in CP-ABE and KP-ABE instances mentioned there, is actually the cost of performing *verifiability-1* whereas we presented the additional cost in case of KP-FE for DFAs is a sort of *public verifiability* to guarantee *verifiability-2* of the underlying CPA-decrypt algorithm (see [11, 13] for these notions of verifiability). Even if, delegation-based conversions are better than verifiability-based conversions in terms of efficiency, there are several schemes for which delegation is still unachieved. One such example is KP-ABE construction of [19] and has been marked NA (i.e. not available) in the above table. Both our direct constructions (Π_R and Π'_R) overcome these problems without affecting the performance. We emphasize that even if delegation is available for some CPA-secure CP-ABE, viz, large universe CP-ABE schemes, the delegation-based conversion is simply not applicable due to efficiency problem. For example, prime order instantiation

[4] The "dark-gray" (resp. "light-gray") has been used to denote *delegation* (resp. *verifiability*)-based construction while parameters and complexity of Π_R and Π'_R (i.e. *direct* constructions) are kept uncolored. Precisely, Π_R is presented at the top among the two uncolored rows for each example. Sometimes, adjacent cells have been merged if corresponding complexities are same. The two-columns under "Decryption Cost" follows the same convention as in Table 1 where the *first* cell is underlying CPA-Decryption cost and *second* cell contains *additional* cost to achieve CCA-Decryption.

([5]) of large universe CP-ABE schemes for [3, Pair Encoding Scheme 13] and dual of [3, Pair Encoding Scheme 4] has transformed key-index size *exponentially big* due to introduction of *dummy attribute* $W = \{0,1\}^{\ell}$.

4 Conclusion

In this work, we presented two direct adaptive CCA-secure predicate encryption constructions to convert adaptive CPA-secure predicate encryption of [5]. The ciphertext of our first construction contains *only one* additional component than in case of adaptive CPA-secure predicate encryption of [5] and decryption needs *exactly three unit* (i.e. $3 \times (d+1)$) additional pairing evaluations. The ciphertext of our second construction, on the other hand, contains *exactly three* additional components (a $G_1^{(d+1)}$ element, an OTSverification key and a signature) than in case of adaptive CPA-secure predicate encryption of [5] and decryption needs *only one unit* additional pairing evaluations. 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

1. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: CRYPTO. Volume 2139 of LNCS., Springer (2001) 213–229
2. Waters, B.: Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In: CRYPTO. Volume 5677 of LNCS., Springer (2009) 619–636
3. Attrapadung, N.: Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In: EUROCRYPT. Volume 8441 of LNCS., Springer (2014) 557–577
4. Wee, H.: Dual system encryption via predicate encodings. In: TCC. Volume 8349 of LNCS., Springer (2014) 455–479
5. Attrapadung, N.: Dual system encryption framework in prime-order groups. In: ASIACRYPT. Volume 10032 of LNCS., Springer (2016) 591–623
6. Chen, J., Gay, R., Wee, H.: Improved dual system ABE in prime-order groups via predicate encodings. In: EUROCRYPT. Volume 9057 of LNCS., Springer (2015) 595–624
7. Chen, J., Wee, H.: Dual system groups and its applications — compact HIBE and more. Cryptology ePrint Archive, Report 2014/265 (2014) `http://eprint.iacr.org/2014/265`.
8. Agrawal, S., Chase, M.: A study of pair encodings: Predicate encryption in prime order groups. In: TCC 2016-A. Volume 9563 of LNCS., Springer (2016) 259–288
9. Agrawal, S., Chase, M.: Simplifying design and analysis of complex predicate encryption schemes. Cryptology ePrint Archive, Report 2017/233 (2017) `http://eprint.iacr.org/2017/233`.
10. Yamada, S., Attrapadung, N., Hanaoka, G., Kunihiro, N.: Generic constructions for chosen-ciphertext secure attribute based encryption. In: PKC. Volume 6571 of LNCS., Springer (2011) 71–89
11. Yamada, S., Attrapadung, N., Santoso, B., Schuldt, J.C.N., Hanaoka, G., Kunihiro, N.: Verifiable predicate encryption and applications to CCA security and anonymous predicate authentication. In: PKC. Volume 7293 of LNCS., Springer (2012) 243–261
12. Nandi, M., Pandit, T.: Generic conversions from CPA to CCA secure functional encryption. Cryptology ePrint Archive, Report 2015/457 (2015) `http://eprint.iacr.org/`.
13. Nandi, M., Pandit, T.: Verifiability-based conversion from CPA to CCA-secure predicate encryption. Journal of Applicable Algebra in Engineering, Communication and Computing (2017) 1–26
14. Blömer, J., Liske, G.: Construction of fully CCA-secure predicate encryptions from pair encoding schemes. In: CT-RSA. Volume 9610 of LNCS., Springer (2016) 431–447
15. Nandi, M., Pandit, T.: On the power of pair encodings: Frameworks for predicate cryptographic primitives. Cryptology ePrint Archive, Report 2015/955 (2015) `http://eprint.iacr.org/2015/955`.
16. Freeman, D.M.: Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In: EUROCRYPT. Volume 6110 of LNCS., Springer (2010) 44–61
17. Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: ACM Conference on Computer and Communications Security. CCS '05, ACM (2005) 320–329
18. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: EUROCRYPT. Volume 3027 of LNCS., Springer (2004) 207–222
19. Lewko, A.B., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In: EUROCRYPT. Volume 6110 of LNCS., Springer (2010) 62–91
20. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for Diffie–Hellman assumptions. Journal of Cryptology **30**(1) (2017) 242–281

A Definitions

A.1 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 $\mathcal{A}$, $\mathsf{Adv}^{\text{sUf-CMA}}_{\mathcal{A},\mathsf{OTS}}(\lambda) \leq \mathsf{neg}(\lambda)$.

A.2 Bilinear Pairing

We consider $(p, G_1, G_2, G_\mathrm{T}, e) \leftarrow \mathcal{G}(\lambda)$ where $\mathcal{G}(\lambda)$ is an asymmetric prime-order bilinear group generator where e is an admissible non-degenerate bilinear map such that g_1 and g_2 are arbitrary generators of G_1 and G_2 respectively. We define $\mathbb{G} = G_1^{d+1}$, $\mathbb{H} = G_2^{d+1}$ and the corresponding pairing operates on $\mathbb{G}$ and $\mathbb{H}$ component-wise and use e to evaluate itself. Precisely, for $g_1^{\mathbf{a}} \in \mathbb{G}$ and $g_2^{\mathbf{b}} \in \mathbb{H}$, $e(g_1^{\mathbf{a}}, g_2^{\mathbf{b}}) = e(g_1, g_2)^{\mathbf{a}^\top \mathbf{b}}$.

A.3 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,

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

where the probability is taken over $(G_1, G_2, G_\mathrm{T}, e, p) \xleftarrow{\$} \mathcal{G}(\lambda)$, $(g_1, g_2) \xleftarrow{\$} G_1 \times G_2$, $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$, $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_p^d$, $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$ and the internal randomness of $\mathcal{A}$ such that $\mathbf{G} = (G_1, G_2, G_\mathrm{T}, e, p, g_1, g_2)$. It is to be noted that *Matrix Diffie-Hellman Problem* is random self reducible [5, 20]. 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 $\left(g_1^{\mathbf{T}}, g_1^{\mathbf{T}\binom{\mathbf{y}}{\hat{y}}}\right)$ it is easy to construct $\left(g_1^{\mathbf{T}}, g_1^{\mathbf{T}\binom{\mathbf{Y}}{\hat{\mathbf{y}}}}\right)$ such that $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_m)$ is uniformly random in $\left(\mathbb{Z}_p^d\right)^m$, $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_m)$ is uniformly random in $(\mathbb{Z}_p)^m$ for $m = \mathsf{poly}(\lambda)$.

A.4 Parameter Hiding Lemma [5, Lemma 2]

Given g_1, g_2, $\mathbf{B}$, $\mathbf{Z}$, $g_1^{\mathbf{W}_i\mathbf{B}\binom{\mathbf{I}_d}{0}} \in G_1^{(d+1)\times d}$ and $g_2^{\mathbf{W}_i^\top\mathbf{Z}\binom{\mathbf{I}_d}{0}} \in G_2^{(d+1)\times d}$, the $(d+1, d+1)^{th}$ entry of the matrix $\mathbf{B}^{-1}\mathbf{W}_i\mathbf{B}$ is information-theoretically hidden where $\mathbf{W}_i \xleftarrow{\$} \mathbb{Z}_p^{(d+1)\times(d+1)}$, $\mathbf{B} \xleftarrow{\$} \mathbb{GL}_{p,d+1}$, $\tilde{\mathbf{D}} \xleftarrow{\$} \mathbb{GL}_{p,d}$ and $\mathbf{Z} = \mathbf{B}^{-\top}\mathbf{D}$ for $\mathbf{D} = \left(\begin{smallmatrix} \tilde{\mathbf{D}} & \mathbf{0} \\ \mathbf{0} & 1 \end{smallmatrix}\right)$.

B CCA-secure Predicate encryption via Conventional approach

The predicate encryption schemes presented in [5] are CPA-secure. We already have pointed out one can convert these schemes to achieve CCA-security by incorporating the generic conversion framework of [10, 11, 13]. Here we show that pair encoding based predicate encryption schemes instantiated in prime-order groups [5] fulfills the notion of *verifiability* [10]. 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 $\tilde{x}$.[5] 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 $\tilde{\mathbf{E}} = \mathsf{Pair}(\tilde{x}, y, N)$.

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

$$\mathsf{Event} = \begin{cases} \prod\limits_{\substack{\iota\in[m_1]\\ \tilde{\iota}\in[w_1]}} e\Big(\mathrm{C}_{\tilde{\iota}}, g_2^{k_\iota(\mathbf{0}, \mathbb{I}_{tj}, \mathbb{W})}\Big)^{\mathbf{E}_{\iota\tilde{\iota}}} = 1 \text{ for all } t\in[d],\ j\in[m_2] & (4) \\ \prod\limits_{\substack{\iota\in[\tilde{m}_1]\\ \tilde{\iota}\in[w_1]}} e\Big(\mathrm{C}_{\tilde{\iota}}, g_2^{\tilde{k}_\iota(\mathbf{0}, \mathbb{I}_{tj}, \mathbb{W})}\Big)^{\tilde{\mathbf{E}}_{\iota\tilde{\iota}}} = 1 \text{ for all } t\in[d],\ j\in[\tilde{m}_2] & (5) \\ \prod\limits_{\substack{\iota\in[m_1]\\ \tilde{\iota}\in[w_1]}} e\big(\mathrm{C}_{\tilde{\iota}}, g_2^{b_\iota \mathbf{1}_t}\big)^{\mathbf{E}_{\iota\tilde{\iota}}} = \prod\limits_{\substack{\iota\in[\tilde{m}_1]\\ \tilde{\iota}\in[w_1]}} e\Big(\mathrm{C}_{\tilde{\iota}}, g_2^{\tilde{b}_\iota \mathbf{1}_t}\Big)^{\tilde{\mathbf{E}}_{\iota\tilde{\iota}}} = e(\mathrm{C}_1, g_2^{\mathbf{1}_t}) \text{ for all } t\in[d]. & (6) \end{cases}$$

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 [5], all the equations $\big($namely Eq. $(4), (5), (6)\big)$ hold true. Note that the correctness of predicate encryption construction of [5] required only Property $\mathcal{P}1$ of regular decryption properties (Sec. 3.1) of underlying pair encoding. However, to satisfy Eq. (6), 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,

$$k_\iota(\boldsymbol{\alpha}, \mathbb{O}, \mathbb{W}) = b_\iota \boldsymbol{\alpha} = \sum_{t\in[d+1]} \alpha_t (b_\iota \mathbf{1}_t), \tag{7}$$

$$k_\iota(\mathbf{0}, \mathbb{R}, \mathbb{W}) = \sum_{\substack{t\in[d]\\ j\in[m_2]}} r_{tj} k_\iota(\mathbf{0}, \mathbb{I}_{tj}, \mathbb{W}). \tag{8}$$

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

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

[5] In conventional approach, one runs $\mathsf{Encrypt}$ and KeyGen on the indexes after running necessary *index-transformation* on them. So here we assume that all the indexes are already transformed accordingly.

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

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

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

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

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

$$= \prod_{\substack{\iota\in[m_1]\\ \tilde{\iota}\in[w_1]}} e\left(\mathrm{C}_{\tilde{\iota}}, g_2^{\sum\limits_{t\in[d+1]} \alpha_t(b_\iota \mathbf{1}_t)}\right)^{\mathbf{E}_{\iota\tilde{\iota}}} \qquad \text{(by Eq. (7))}$$

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

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

$$= \prod_{t\in[d+1]} \left(e(\mathrm{C}_1, g_2^{\mathbf{1}_t})\right)^{\alpha_t} \qquad \text{(by Eq. (6))}$$

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

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

$$= \prod_{\substack{\iota\in[m_1]\\ \tilde{\iota}\in[w_1]}} e\left(\mathrm{C}_{\tilde{\iota}}, g_2^{\sum\limits_{\substack{t\in[d]\\ j\in[m_2]}} r_{tj} k_\iota(\mathbf{0},\mathbb{I}_{tj},\mathbb{W})}\right)^{\mathbf{E}_{\iota\tilde{\iota}}} \qquad \text{(by Eq. (8))}$$

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

$$= \prod_{\substack{t\in[d]\\ j\in[m_2]}} (1)^{r_{tj}} = 1. \qquad \text{(by Eq. (4))}$$

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

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

Hence we note that [5] schemes achieves *verifiability* and can be converted generically to achieve CCA-security [10, 11, 13]. 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 6. 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 $\big(I \times (m_2+1) + \tilde{I} \times (\tilde{m}_2+1)+1\big) \times d \times (d+1)$ 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 $3(d+1)$ (Π_R in Section 3.2) and $(d+1)$ (Π'_R in Section 3.4) additional pairings only.

C Rest of Proof of Theorem 1

Here we recall the indistinguishability of games that we mimic from [5] after appropriate modification. Note that, here we discuss these games for the proof of Theorem 1. All the games discussed in this section, can be easily ported for the proof of Theorem 2.

C.1 Normal to Semi-functional Ciphertext

Lemma 6 (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 decryption queries, there exists a PPT algorithm $\mathcal{B}_1$ such that $|\mathsf{Adv}^0_{\mathcal{A}}(\lambda) - \mathsf{Adv}^1_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\mathcal{D}_d\text{-MatDH}}_{\mathcal{B}_1}(\lambda)$.*

Proof. The algorithm $\mathcal{B}_1$ gets input $(\mathbf{G}, g_1^{\mathbf{T}}, g_1^{\mathbf{T}\binom{\mathbf{y}}{\hat{y}}})$ as $\mathcal{D}_d$-MatDH problem instance where $\hat{y} = 0$ or $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$, $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_p^d$.

Setup. $\mathcal{B}_1$ 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

$$\mathbf{D} = \mathbf{B}^{\top}\mathbf{Z} = \begin{pmatrix} \mathbf{T}^{\top} & \tilde{\mathbf{B}}^{\top} \end{pmatrix} \left(\tilde{\mathbf{B}}^{-\top} \begin{pmatrix} \mathbf{J} & -\mathbf{M}^{-\top}\mathbf{c}^{\top} \\ \mathbf{0} & 1 \end{pmatrix} \right) = \begin{pmatrix} \mathbf{M}^{\top} & \mathbf{c}^{\top} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{J} & -\mathbf{M}^{-\top}\mathbf{c}^{\top} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{M}^{\top}\mathbf{J} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix}.$$

$\mathcal{B}_1$ can compute $g_1^{\mathbf{B}} = g_1^{\tilde{\mathbf{B}}\mathbf{T}}$ and $g_2^{\mathbf{Z}\binom{\mathbf{I}_d}{0}} = g_2^{\tilde{\mathbf{B}}^{-\top}\binom{\mathbf{J}}{0}}$. Therefore it can easily compute mpk, msk by choosing the parameters $\boldsymbol{\alpha}, \mathbf{W}_1, \ldots, \mathbf{W}_{n+2}$ 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 $\mathcal{B}_1$ computes normal altKey $\hat{\mathbf{K}} \leftarrow \mathsf{AltKeyGen}(\overline{\mathbf{C}}, x, msk)$ and returns $\mathsf{AltDecrypt}(\overline{\mathbf{C}}, \hat{\mathbf{K}})$ to $\mathcal{A}$.

Challenge. On receiving the challenge (y^*, M_0, M_1), $\mathcal{B}_1$ 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 $(\mathbf{G}, g_1^{\mathbf{T}}, g_1^{\mathbf{T}\binom{\mathbf{Y}}{\hat{\mathbf{y}}}})$. The decision problem is now to find if $\hat{\mathbf{y}} = \mathbf{0}$ or $\hat{\mathbf{y}} \xleftarrow{\$} (\mathbb{Z}_p)^{(w_2+1)}$ where $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$ and $\mathbf{Y} \xleftarrow{\$} (\mathbb{Z}_p^d)^{(w_2+1)}$. $\mathcal{B}_1$ implicitly sets $\binom{\mathbf{Y}}{\hat{\mathbf{y}}} = \mathbf{S} + \hat{S} = \begin{pmatrix} \mathbf{s}_0 & \cdots & \mathbf{s}_{w_2} \\ \hat{s}_0 & \cdots & \hat{s}_{w_2} \end{pmatrix}$. As $\mathcal{B}_1$ has $g_1^{\mathbf{T}\binom{\mathbf{Y}}{\hat{\mathbf{y}}}}$, it can compute $g_1^{\mathbf{B}\binom{\mathbf{s}_j}{\hat{s}_j}} = g_1^{\tilde{\mathbf{B}}\mathbf{T}\binom{\mathbf{s}_j}{\hat{s}_j}} = g_1^{\tilde{\mathbf{B}}\mathbf{T}\binom{\mathbf{y}_j}{\hat{y}_j}}$ for $j \in [0, w_2]$. As $\mathcal{B}_1$ knows $\boldsymbol{\alpha}$, $\mathbf{W}_1, \ldots, \mathbf{W}_{n+2}$, it can compute all components of ciphertext.

Then $\mathcal{B}_1$ computes $\mathbf{C}^*$ as it knows $\boldsymbol{\alpha}$, $\mathbf{W}_1, \ldots, \mathbf{W}_{n+2}$. Then it evaluates $\xi^* = \mathcal{H}(\mathbf{C}^*)$ to compute $\overline{\mathbf{C}}^* = (\overline{\mathrm{C}}^*_0, \mathbf{C}^*)$ where $\overline{\mathrm{C}}^*_0 = g_1^{(\xi^*\mathbf{W}_{n+1} + \mathbf{W}_{n+2})\mathbf{B}\binom{\mathbf{s}_0}{\hat{s}_0}}$. 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 $\mathfrak{b}'$. $\mathcal{B}_1$ outputs 1 if $\mathfrak{b}' = \mathfrak{b}$ and 0 otherwise.

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

Lemma 7 ($\mathsf{Game}_{2,i-1,3}$ **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 decryption queries, there exists a PPT algorithm $\mathcal{B}_1$ such that $|\mathsf{Adv}^{2,i-1,3}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{2,i,1}_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\mathcal{D}_d\text{-MatDH}}_{\mathcal{B}_1}(\lambda)$.*

Proof. The algorithm $\mathcal{B}_1$ gets as input $(\mathbf{G}, g_2^{\mathbf{T}}, g_2^{\mathbf{T}\binom{\mathbf{y}}{\hat{y}}})$ as $\mathcal{D}_d$-MatDH problem instance where $\hat{y} = 0$ or $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$, $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_p^d$.

Setup. $\mathcal{B}_1$ chooses $\tilde{\mathbf{B}} \xleftarrow{\$} \mathbb{GL}_{p,d+1}, \mathbf{J} \xleftarrow{\$} \mathbb{GL}_{p,d}$ and sets

$$\mathbf{B} = \tilde{\mathbf{B}} \begin{pmatrix} \mathbf{I}_d & \mathbf{M}^{-\top}\mathbf{c}^{\top} \\ \mathbf{0} & -1 \end{pmatrix} \text{ and } \mathbf{D} = \begin{pmatrix} \mathbf{MJ} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} \text{ where } \mathbf{T} = \begin{pmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{c} & 1 \end{pmatrix} \text{ due to } \mathcal{D}_d\text{-MatDH assumption.}$$

Then it defines

$$\mathbf{Z} = \mathbf{B}^{-\top}\mathbf{D} = \tilde{\mathbf{B}}^{-\top} \begin{pmatrix} \mathbf{I}_d & \mathbf{0} \\ \mathbf{c}\mathbf{M}^{-1} & -1 \end{pmatrix} \begin{pmatrix} \mathbf{MJ} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} = \tilde{\mathbf{B}}^{-\top}\mathbf{T} \begin{pmatrix} \mathbf{J} & \mathbf{0} \\ \mathbf{0} & -1 \end{pmatrix}.$$

Then define $\tilde{\mathbf{Z}} = \begin{pmatrix} \mathbf{J} & \mathbf{0} \\ \mathbf{0} & -1 \end{pmatrix}$ so that $\mathbf{Z} = \tilde{\mathbf{B}}^{-\top}\mathbf{T}\tilde{\mathbf{Z}}$. $\mathcal{B}_1$ therefore can compute the public parameters as $g_1^{\mathbf{B}\binom{\mathbf{I}_d}{0}} = g_1^{\tilde{\mathbf{B}}\binom{\mathbf{I}_d}{0}}$ and $g_2^{\mathbf{Z}} = g_2^{\tilde{\mathbf{B}}^{-\top}\mathbf{T}\tilde{\mathbf{Z}}}$. Then $\mathcal{B}_1$ chooses $\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)}$ and $\mathbb{W} = (\mathbf{W}_1, \ldots, \mathbf{W}_{n+2}) \xleftarrow{\$} \left(\mathbb{Z}_p^{(d+1)\times(d+1)}\right)^{(n+2)}$ and publishes public key mpk. Note that $\mathcal{B}_1$ 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 ($j \leq q_1$), outputs secret key $\mathbf{K}$ as follows.

- If $j > i$, $\mathcal{B}_1$ generates normal key $\mathsf{KeyGen}(x, msk)$.
- If $j < i$, $\mathcal{B}_1$ generates type-3 key $\mathsf{SFKeyGen}(x, msk, -, \widehat{mpk}_{base}, 3, \beta_j)$ for $\beta_j \xleftarrow{\$} \mathbb{Z}_p$.
- If $j = i$, $\mathcal{B}_1$ 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 $(\mathbf{G}, g_2^{\mathbf{T}}, g_2^{\mathbf{T}\binom{\mathbf{Y}}{\hat{\mathbf{y}}}})$. The decision problem is now to find if $\hat{\mathbf{y}} = \mathbf{0}$ or $\hat{\mathbf{y}} \xleftarrow{\$} (\mathbb{Z}_p)^{m_2}$ where $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$ and $\mathbf{Y} \xleftarrow{\$} \left(\mathbb{Z}_p^d\right)^{m_2}$. $\mathcal{B}_1$ implicitly sets $\tilde{\mathbf{Z}}^{-1}\binom{\mathbf{Y}}{\hat{\mathbf{y}}} = \mathbf{R} + \hat{R} = \begin{pmatrix} \mathbf{r}_1 \cdots \mathbf{r}_{m_2} \\ \hat{r}_1 \cdots \hat{r}_{m_2} \end{pmatrix}$. Therefore $g_2^{\mathbf{Z}\binom{\mathbf{r}_j}{\hat{r}_j}} = g_2^{\left(\tilde{\mathbf{B}}^{-\top}\mathbf{T}\tilde{\mathbf{Z}}\right)\left(\tilde{\mathbf{Z}}^{-1}\binom{\mathbf{y}_j}{\hat{y}_j}\right)}$. As $\mathcal{B}_1$ has $g_2^{\mathbf{T}\binom{\mathbf{Y}}{\hat{\mathbf{y}}}}, \boldsymbol{\alpha}, \tilde{\mathbf{B}}, \mathbf{W}_1, \ldots, \mathbf{W}_{n+2}$, it can compute all components of secret key. It is evident from the description that if $\hat{\mathbf{y}} = \mathbf{0}$, the key is a normal key whereas if $\hat{\mathbf{y}} \xleftarrow{\$} (\mathbb{Z}_p)^{m_2}$, 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 $\mathcal{B}_1$ computes normal altKey $\hat{\mathbf{K}} \leftarrow \mathsf{AltKeyGen}(\overline{\mathbf{C}}, x, msk)$ and returns $\mathsf{SFAltDecrypt}(\overline{\mathbf{C}}, \hat{\mathbf{K}})$ to $\mathcal{A}$.

Challenge. On receiving the challenge (y^*, M_0, M_1), $\mathcal{B}_1$ 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 $\binom{\mathbf{s}'_j}{\hat{s}'_j} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)}$ and implicitly sets $\binom{\mathbf{s}_j}{\hat{s}_j} = \mathbf{B}^{-1}\binom{\mathbf{s}'_j}{\hat{s}'_j}$.

Then $\mathcal{B}_1$ computes $\mathbf{C}^*$ as it knows $\boldsymbol{\alpha}$, $\mathbf{W}_1, \ldots, \mathbf{W}_{n+2}$. Then it evaluates $\xi^* = \mathcal{H}(\mathbf{C}^*)$ to compute $\overline{\mathbf{C}}^* = (\overline{\mathrm{C}}_0^*, \mathbf{C}^*)$ where $\overline{\mathrm{C}}_0^* = g_1^{(\xi^*\mathbf{W}_{n+1} + \mathbf{W}_{n+2})\binom{\mathbf{s}'_0}{\hat{s}'_0}}$. It outputs $\overline{\mathbf{C}}^*$.

Key Queries. On j^{th} secret key query x ($j \in [q_1 + 1, q]$), $\mathcal{B}_1$ generates normal key $\mathsf{KeyGen}(x, msk)$.

Dec Queries. Same as Phase-I decryption queries.

Guess. $\mathcal{A}$ halts with output $\mathfrak{b}'$. $\mathcal{B}_1$ outputs 1 if $\mathfrak{b}' = \mathfrak{b}$ and 0 otherwise.

C.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 $\mathsf{SFEncrypt}$, $\mathsf{SFKeyGen}$ and $\mathsf{SFAltKeyGen}$.

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

– $\mathsf{SFKeyGen}(x, msk, \widehat{mpk}_z, \widehat{mpk}_{base}, \mathrm{type}, \beta)$: 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

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

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

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

– $\mathsf{SFEncrypt}(y, M, mpk, \widehat{mpk}_b)$: 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 $\widehat{\mathbf{S}} = \left(\left(\begin{smallmatrix}\mathbf{0}\\\hat{s}_0\end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix}\mathbf{0}\\\hat{s}_{w_2}\end{smallmatrix}\right) \right) \in \left(\mathbb{Z}_p^{(d+1)}\right)^{(w_2+1)}$. Computes semi-functional ciphertext $\mathbf{C} = (\mathrm{C}_1, \ldots, \mathrm{C}_{w_1}, \mathrm{C}_{w_1+1})$ where for $\tilde{\iota} \in [w_1]$ each

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

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

– $\mathsf{SFAltKeyGen}(\overline{\mathbf{C}}, x, msk, \widehat{mpk}_z, \widehat{mpk}_{base}, \mathrm{type}, \eta)$: 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 $\mathbf{K} = \left\{ g_2^{k_\iota(\boldsymbol{\alpha},\mathbf{R},\mathbb{W})} \right\}_{\iota\in[m_1]} \in \left(G_2^{(d+1)}\right)^{m_1}$ where each

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

Then it computes $(\tilde{\mathrm{K}}_1, \ldots, \tilde{\mathrm{K}}_{w_1})$ where $\tilde{\mathrm{K}}_{\tilde{\iota}} = \prod_{\iota\in[m_1]} (\mathrm{K}_\iota)^{\mathbf{E}_{\iota\tilde{\iota}}}$ for each $\tilde{\iota} \in [w_1]$.
Defines modified key $\hat{\mathrm{K}} = (\hat{\mathrm{K}}_0, \Phi \cdot \tilde{\mathrm{K}}_1, \tilde{\mathrm{K}}_2, \ldots, \tilde{\mathrm{K}}_{w_1})$ where

$$(\hat{\mathrm{K}}_0, \Phi) = \begin{cases} \left(g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\\hat{\mathfrak{r}}\end{smallmatrix}\right)}, g_2^{(\xi\mathbf{W}_{n+1}^\top+\mathbf{W}_{n+2}^\top)\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\\hat{\mathfrak{r}}\end{smallmatrix}\right)+\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\(\xi\hat{\mathrm{w}}_{n+1}+\hat{\mathrm{w}}_{n+2})\hat{\mathfrak{r}}\end{smallmatrix}\right)} \right) & \text{if type} = 1, \\ \left(g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\\hat{\mathfrak{r}}\end{smallmatrix}\right)}, g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\\eta u\end{smallmatrix}\right)+(\xi\mathbf{W}_{n+1}^\top+\mathbf{W}_{n+2}^\top)\mathbf{Z}\left(\begin{smallmatrix}\mathfrak{r}\\\hat{\mathfrak{r}}\end{smallmatrix}\right)+\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\(\xi\hat{\mathrm{w}}_{n+1}+\hat{\mathrm{w}}_{n+2})\hat{\mathfrak{r}}\end{smallmatrix}\right)} \right) & \text{if type} = 2, \end{cases}$$

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

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 $\mathsf{SFKeyGen}$, $\mathsf{SFEncrypt}$ and $\mathsf{SFAltKeyGen}$ to generate the secret keys, the challenge ciphertext and the altKeys containing the randomness newly introduced via $\hat{\mathbf{w}}$. This result will be used in arguing indistinguishibility of type-1 and type-2 keys of both secret keys and altKeys (Lemma 3, Lemma 8 and Lemma 11).

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

Proof. Due to parameter-hiding lemma in Section A.4, $\mathbf{W}_i \xleftarrow{\$} \mathbb{Z}_p^{(d+1)\times(d+1)}$, $\mathbf{B} \xleftarrow{\$} \mathbb{GL}_{p,d+1}$ and $\hat{\mathrm{w}}_i \xleftarrow{\$} \mathbb{Z}_p$, both $R_{\mathbf{W}_i,\hat{\mathrm{w}}_i}$ and $\mathbf{W}_i$ are identically distributed where $R_{\mathbf{W}_i,\hat{\mathrm{w}}_i} = \mathbf{W}_i + \mathbf{B}\begin{pmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \hat{\mathrm{w}}_i\end{pmatrix}\mathbf{B}^{-1}$ for $i \in [n+2]$. It can easily be verified that for each $i \in [n+2]$, $R_{\mathbf{W}_i,\hat{\mathrm{w}}_i}\mathbf{B}\left(\begin{smallmatrix}\mathbf{I}_d\\ 0\end{smallmatrix}\right) = \mathbf{W}_i\mathbf{B}\left(\begin{smallmatrix}\mathbf{I}_d\\ 0\end{smallmatrix}\right)$, $R^\top_{\mathbf{W}_i,\hat{\mathrm{w}}_i}\mathbf{Z}\left(\begin{smallmatrix}\mathbf{I}_d\\ 0\end{smallmatrix}\right) = \mathbf{W}_i^\top\mathbf{Z}\left(\begin{smallmatrix}\mathbf{I}_d\\ 0\end{smallmatrix}\right)$, $R_{\mathbf{W}_i,\hat{\mathrm{w}}_i}\mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\ 1\end{smallmatrix}\right) = \mathbf{W}_i\mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\ 1\end{smallmatrix}\right) + \mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\ \hat{\mathrm{w}}_i\end{smallmatrix}\right)$ and $R^\top_{\mathbf{W}_i,\hat{\mathrm{w}}_i}\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ 1\end{smallmatrix}\right) = \mathbf{W}_i^\top\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ 1\end{smallmatrix}\right) + \mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ \hat{\mathrm{w}}_i\end{smallmatrix}\right)$. 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.

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

Lemma 8 ($\mathsf{Game}_{2,i,1}$ **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 decryption queries, there exists a PPT algorithm $\mathcal{B}_2$ such that $|\mathsf{Adv}^{2,i,1}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{2,i,2}_{\mathcal{A}}(\lambda)| \le \mathsf{Adv}^{\mathsf{CMH}}_{\mathcal{B}_2}(\lambda)$.*

Proof. In this co-selective security game of pair encoding scheme, the algorithm $\mathcal{B}_2$ gets as input the group description G_1, G_2, G_{T}, $g_1 \in G_1$ and $g_2 \in G_2$.

Setup. $\mathcal{B}_2$ chooses $\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)}$, $\mathbb{W} = (\mathbf{W}_1, \ldots, \mathbf{W}_{n+2}) \xleftarrow{\$} \left(\mathbb{Z}_p^{(d+1)\times(d+1)}\right)^{(n+2)}$, $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)\times(d+1)}$, $\tilde{\mathbf{D}} \xleftarrow{\$} \mathbb{GL}_{p,d}$ 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 ($j \le q_1$), outputs secret key $\mathbf{K}$ as follows.

- If $j > i$, $\mathcal{B}_2$ generates normal key $\mathsf{KeyGen}(x, msk)$.
- If $j < i$, $\mathcal{B}_2$ generates type-3 key $\mathsf{SFKeyGen}(x, msk, -, \widehat{mpk}_{base}, 3, \beta_j)$ after choosing $\beta_j \xleftarrow{\$} \mathbb{Z}_p$.
- If $j = i$, $\mathcal{B}_2$ forwards x as the challenge query to the challenger to receive $\mathbf{V} = g_2^{\mathbf{k}(\beta,\hat{\mathbf{r}},\hat{\mathbf{w}})}$ where $(\mathbf{k} = (k_1, \ldots, k_{m_1}); m_2) \leftarrow \mathsf{EncK}(x, N)$. $\mathcal{B}_2$ has to decide if $\beta = 0$ or $\beta \xleftarrow{\$} \mathbb{Z}_p$. It is to be noted that $\hat{\mathbf{r}}$ and $\hat{\mathbf{w}}$ are chosen by the challenger of CMH-*Security* game, unknown to $\mathcal{B}_2$. Now $\mathcal{B}_2$ computes the normal part of the key by computing $\left\{g_2^{k_\iota(\boldsymbol{\alpha},\mathbf{R},\mathbb{W})}\right\}_{\iota\in[m_1]}$ 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 $\hat{\mathbf{w}}$ which is unknown to $\mathcal{B}_2$, it implicitly sets $\hat{R} = \left(\left(\begin{smallmatrix}\mathbf{0}\\ \hat{r}_1\end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix}\mathbf{0}\\ \hat{r}_{m_2}\end{smallmatrix}\right)\right)$ where $\hat{\mathbf{r}} = (\hat{r}_1, \ldots, \hat{r}_{m_2})$. Then the semi-functional component of the key is
$$g_2^{\mathbf{k}'(\beta_i,\hat{R},\mathbb{W},\hat{\mathbf{w}})} = \left\{ g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ k_\iota(\beta,\hat{\mathbf{r}},\hat{\mathbf{w}})\end{smallmatrix}\right)} \prod_{\substack{j\in[m_2]\\ k\in[n]}} g_2^{b_{\iota jk}\mathbf{W}_k^\top\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ \hat{r}_j\end{smallmatrix}\right)} \right\}_{\iota\in[m_1]}.$$
Notice that $\mathcal{B}_2$ implicitly sets β_i to be β that is actually set by the challenger of CMH-*Security* game and unknown to $\mathcal{B}_2$. Since $\mathcal{B}_2$ 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. $g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ k_\iota(\beta,\hat{\mathbf{r}},\hat{\mathbf{w}})\end{smallmatrix}\right)} = V_\iota^{\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ 1\end{smallmatrix}\right)}$ for $\iota \in [m_1]$.

However to compute the second component $\left(\prod_{\substack{j\in[m_2]\\k\in[n]}} g_2^{b_{\iota jk}\mathbf{W}_k^\top \mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ \hat{r}_j\end{smallmatrix}\right)}\right)$ of the semi-functional part of the secret key, $\mathcal{B}_2$ 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 $k_{\iota'}(\beta, \hat{\mathbf{r}}, \hat{\mathbf{w}}) = \hat{r}_j$, that lets $\mathcal{B}_2$ to know $\hat{r}_j$.
- Or there is no such $\iota' \in [m_1]$ for which $k_{\iota'}(\beta, \hat{\mathbf{r}}, \hat{\mathbf{w}}) = \hat{r}_j$. Then due to regular decryption properties of pair encoding (precisely Property $\mathcal{P}2$), $b_{\iota'' jk} = 0$ for all $\iota'' \in [m_1]$, $k \in [n]$.

$\mathcal{B}_2$ 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 $\mathcal{B}_2$ computes normal altKey $\hat{\mathbf{K}} \leftarrow \mathsf{AltKeyGen}(\overline{\mathbf{C}}, x, msk)$ and returns $\mathsf{AltDecrypt}(\overline{\mathbf{C}}, \hat{\mathbf{K}})$ to $\mathcal{A}$.

Challenge. On receiving the challenge (y^*, M_0, M_1), $\mathcal{B}_2$ 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. $\mathcal{B}_2$ receives $\mathbf{U} \leftarrow g_1^{\mathbf{c}(\hat{\mathbf{s}}, \hat{\mathbf{w}})}$.

$\mathcal{B}_2$ first computes the normal part of the ciphertext by computing $g_1^{c_{\tilde{\iota}}(\mathbf{S}, \mathbb{W})}$ 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 $\hat{\mathbf{w}}$ which is unknown to $\mathcal{B}_2$, it implicitly sets $\hat{S} = \left(\left(\begin{smallmatrix}\mathbf{0}\\ \hat{s}_0\end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix}\mathbf{0}\\ \hat{s}_{w_2}\end{smallmatrix}\right)\right)$ where $\hat{\mathbf{s}} = (\hat{s}_0, \ldots, \hat{s}_{w_2})$. Then it computes the semi-functional component of the ciphertext as

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

Since $\mathcal{B}_2$ 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. $g_1^{\mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\ c_{\tilde{\iota}}(\hat{\mathbf{s}}, \hat{\mathbf{w}})\end{smallmatrix}\right)} = U_{\tilde{\iota}}^{\mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\ 1\end{smallmatrix}\right)}$.

However to compute the second component $\left(\prod_{\substack{j\in[0,w_2]\\k\in[n]}} g_1^{a_{\tilde{\iota} jk}\mathbf{W}_k\mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\ \hat{s}_j\end{smallmatrix}\right)}\right)$ of the semi-functional part of the ciphertext, $\mathcal{B}_2$ 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 $c_{\tilde{\iota}'}(\hat{\mathbf{s}}, \hat{\mathbf{w}}) = \hat{s}_j$, that lets $\mathcal{B}_2$ to know $\hat{s}_j$.
- Or there is no such $\tilde{\iota}' \in [w_1]$ for which $c_{\tilde{\iota}'}(\hat{\mathbf{s}}, \hat{\mathbf{w}}) = \hat{s}_j$. 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 $\mathcal{P}4$), $\mathcal{B}_2$ 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 $\mathbf{U}$.

$\mathcal{B}_2$ uses the normal part of the ciphertext and the semi-functional part of the ciphertext to generate the $\mathbf{C}^*$. Then it evaluates $\xi^* = \mathcal{H}(\mathbf{C}^*)$ to compute $\overline{\mathrm{C}}_0^*$ and defines $\overline{\mathbf{C}}^* = (\overline{\mathrm{C}}_0^*, \mathbf{C}^*)$. It outputs $\overline{\mathbf{C}}^*$.

Key Queries. On j^{th} secret key query x ($j \in [q_1 + 1, q]$), $\mathcal{B}_2$ generates secret key $\mathbf{K} \leftarrow \mathsf{KeyGen}(x, msk)$.

Dec Queries. Same as Phase-I decryption queries.

Guess. $\mathcal{A}$ halts with output $\mathfrak{b}'$. $\mathcal{B}_2$ outputs 1 if $\mathfrak{b}' = \mathfrak{b}$ and 0 otherwise.

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

Lemma 9 ($\mathsf{Game}_{2,i,2}$ **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 decryption queries, there exists a PPT algorithm $\mathcal{B}_1$ such that $|\mathsf{Adv}^{2,i,2}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{2,i,3}_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\mathcal{D}_d\text{-MatDH}}_{\mathcal{B}_1}(\lambda)$.*

Proof. The algorithm $\mathcal{B}_1$ gets as input $(\mathbf{G}, g_2^{\mathbf{T}}, g_2^{\mathbf{T}\binom{\mathbf{y}}{\hat{y}}})$ as $\mathcal{D}_d$-MatDH problem instance where $\hat{y} = 0$ or $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$, $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_p^d$.

The simulator description is same as Lemma 7 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 $g_2^{k_\iota(\mathbf{Z}\binom{\mathbf{0}}{\beta_i}, \mathbf{0}, \mathbb{W})} \in G_2^{(d+1)}$. As $\mathcal{B}_1$ knows $\widehat{mpk}_{\text{base}}$, it chooses $\beta_i \xleftarrow{\$} \mathbb{Z}_p$ to perform the simulation. In the similar light of Lemma 7, we see that if $\hat{\mathbf{y}} = \mathbf{0}$, the key is a type-3 key whereas if $\hat{\mathbf{y}} \xleftarrow{\$} (\mathbb{Z}_p)^{m_2}$, the key is type-2 key.

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

Lemma 10 ($\mathsf{Game}_{2,q_1,3}$ **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 decryption queries, there exists a PPT algorithm $\mathcal{B}_1$ such that $|\mathsf{Adv}^{2,q_1,3}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{2,q_1+1,1}_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\mathcal{D}_d\text{-MatDH}}_{\mathcal{B}_1}(\lambda)$.*

Proof. The algorithm $\mathcal{B}_1$ gets as input $(\mathbf{G}, g_2^{\mathbf{T}}, g_2^{\mathbf{T}\binom{\mathbf{y}}{\hat{y}}})$ as $\mathcal{D}_d$-MatDH problem instance where $\hat{y} = 0$ or $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$, $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_p^d$.

The simulator description is same as Lemma 7 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 7, we see that if $\hat{\mathbf{y}} = \mathbf{0}$, the key is a normal key whereas if $\hat{\mathbf{y}} \xleftarrow{\$} (\mathbb{Z}_p)^{m_2}$, the key is type-1 key.

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

Lemma 11 ($\mathsf{Game}_{2,q_1+1,1}$ **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 decryption queries, there exists a PPT algorithm $\mathcal{B}_3$ such that $|\mathsf{Adv}^{2,q_1+1,1}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{2,q_1+1,2}_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\mathsf{SMH}}_{\mathcal{B}_3}(\lambda)$.*

Proof. In this selective security game of pair encoding scheme, the algorithm $\mathcal{B}_3$ gets as input the group description G_1, G_2, G_{T}, $g_1 \in G_1$ and $g_2 \in G_2$.

Setup. $\mathcal{B}_3$ chooses $\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)}$ and $\mathbb{W} = (\mathbf{W}_1, \ldots, \mathbf{W}_{n+2}) \xleftarrow{\$} \left(\mathbb{Z}_p^{(d+1)\times(d+1)}\right)^{(n+2)}$, $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)\times(d+1)}$, $\tilde{\mathbf{D}} \xleftarrow{\$} \mathbb{GL}_{p,d}$ 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 ($j \leq q_1$), $\mathcal{B}_3$ 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 $\mathcal{B}_3$ computes normal altKey $\hat{\mathbf{K}} \leftarrow \mathsf{AltKeyGen}(\overline{\mathbf{C}}, x, msk)$ and returns $\mathsf{AltDecrypt}(\overline{\mathbf{C}}, \hat{\mathbf{K}})$ to $\mathcal{A}$.

Challenge. On receiving the challenge (y^*, M_0, M_1), $\mathcal{B}_3$ 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. $\mathcal{B}_3$ receives $\mathbf{U} \leftarrow g_1^{\mathbf{c}(\hat{\mathbf{s}}, \hat{\mathbf{w}})}$.

$\mathcal{B}_3$ first computes the normal part of the ciphertext by computing $g_1^{c_{\tilde{\iota}}(\mathbf{S},\mathbb{W})}$ for $\mathbf{S} = \left(\binom{\mathbf{s}_0}{0}, \ldots, \binom{\mathbf{s}_{w_2}}{0}\right)$ such that $\mathbf{s}_0, \ldots, \mathbf{s}_{w_2} \xleftarrow{\$} \mathbb{Z}_p^d$.

To compute the semi-functional part, that contains $\hat{\mathbf{w}}$ which is unknown to $\mathcal{B}_3$, it implicitly sets $\hat{S} = \left(\binom{\mathbf{0}}{\hat{s}_0}, \ldots, \binom{\mathbf{0}}{\hat{s}_{w_2}}\right)$ where $\hat{\mathbf{s}} = (\hat{s}_0, \ldots, \hat{s}_{w_2})$. Then it computes the semi-functional component of the ciphertext as

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

Since $\mathcal{B}_3$ 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. $g_1^{\mathbf{B}\binom{\mathbf{0}}{c_{\tilde{\iota}}(\hat{\mathbf{s}},\hat{\mathbf{w}})}} = U_{\tilde{\iota}}^{\mathbf{B}\binom{\mathbf{0}}{1}}$.

However to compute the second component $\left(\prod_{\substack{j\in[0,w_2]\\ k\in[n]}} g_1^{a_{\tilde{\iota}jk}\mathbf{W}_k\mathbf{B}\binom{\mathbf{0}}{\hat{s}_j}}\right)$ of the semi-functional part of the challenge ciphertext, $\mathcal{B}_3$ 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 $c_{\tilde{\iota}'}(\hat{\mathbf{s}}, \hat{\mathbf{w}}) = \hat{s}_j$, that lets $\mathcal{B}_3$ to know such an $\hat{s}_j$.
- Or there is no such $\tilde{\iota}' \in [w_1]$ for which $c_{\tilde{\iota}'}(\hat{\mathbf{s}}, \hat{\mathbf{w}}) = \hat{s}_j$. 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 $\mathcal{P}4$), $\mathcal{B}_2$ also can compute the semi-functional component of the blinding factor $e(g_1^{\boldsymbol{\alpha}^\top\mathbf{B}\binom{\mathbf{0}}{\hat{s}_0}}, g_2)$ as $g_1^{\hat{s}_0}$ is available in $\mathbf{U}$.

$\mathcal{B}_3$ uses the normal part of the ciphertext and the semi-functional part of the ciphertext to generate the $\mathbf{C}^*$. Then it evaluates $\xi^* = \mathcal{H}(\mathbf{C}^*)$ to compute $\overline{\mathrm{C}}_0^*$ and defines $\overline{\mathbf{C}}^* = (\overline{\mathrm{C}}_0^*, \mathbf{C}^*)$. It outputs $\overline{\mathbf{C}}^*$.

Key Queries. On j^{th} secret key query x_j ($j \in [q_1+1, q]$) $\mathcal{B}_3$ forwards x_j as a key-query to the challenger to receive $\mathbf{V} = g_2^{\mathbf{k}(\beta,\hat{\mathbf{r}},\hat{\mathbf{w}})}$ where $(\mathbf{k} = (k_1, \ldots, k_{m_1}); m_2) \leftarrow \mathsf{EncK}(x_j, N)$. $\mathcal{B}_3$ has to decide if $\beta = 0$ or $\beta \xleftarrow{\$} \mathbb{Z}_p$. It is to be noted that $\hat{r}$ and $\hat{\mathbf{w}}$ are chosen by the challenger of SMH-*Security* game, unknown to $\mathcal{B}_3$. So $\mathcal{B}_3$ computes the normal part of the key by computing $g_2^{k_\iota(\boldsymbol{\alpha},\mathbf{R},\mathbb{W})}$ for $\mathbf{R} = \left(\binom{\mathbf{r}_1}{0}, \ldots, \binom{\mathbf{r}_{m_2}}{0}\right)$ such that $\mathbf{r}_1, \ldots, \mathbf{r}_{m_2} \xleftarrow{\$} \mathbb{Z}_p^d$. To compute the semi-functional part, that contains $\hat{\mathbf{w}}$ which is unknown to $\mathcal{B}_3$, it implicitly sets $\hat{R} = \left(\binom{\mathbf{0}}{\hat{r}_1}, \ldots, \binom{\mathbf{0}}{\hat{r}_{m_2}}\right)$ where $\hat{\mathbf{r}} = (\hat{r}_1, \ldots, \hat{r}_{m_2})$. Then it computes the semi-functional component of the key as following.

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

Since $\mathcal{B}_3$ 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. $g_2^{\mathbf{Z}\binom{\mathbf{0}}{k_\iota(\beta,\hat{\mathbf{r}},\hat{\mathbf{w}})}} = V_\iota^{\mathbf{Z}\binom{\mathbf{0}}{1}}$.

However to compute the second component $\left(\prod_{\substack{j\in[m_2]\\ k\in[n]}} g_2^{b_{\iota jk}\mathbf{W}_k^\top\mathbf{Z}\binom{\mathbf{0}}{\hat{r}_j}}\right)$ of the semi-functional part of the secret key, $\mathcal{B}_3$ 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 $k_{\iota'}(\beta, \hat{\mathbf{r}}, \hat{\mathbf{w}}) = \hat{r}_j$, that lets $\mathcal{B}_3$ to know $\hat{r}_j$.
- Or there is no such $\iota' \in [m_1]$ for which $k_{\iota'}(\beta, \hat{\mathbf{r}}, \hat{\mathbf{w}}) = \hat{r}_j$. Then due to regular decryption properties of pair encoding (precisely Property $\mathcal{P}2$), $b_{\iota''jk} = 0$ for all $\iota'' \in [m_1]$, $k \in [n]$.

$\mathcal{B}_3$ 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 $\mathfrak{b}'$. $\mathcal{B}_3$ outputs 1 if $\mathfrak{b}' = \mathfrak{b}$ and 0 otherwise.

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

Lemma 12 ($\mathsf{Game}_{2,q_1+1,2}$ **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 decryption queries, there exists a PPT algorithm $\mathcal{B}_1$ such that $|\mathsf{Adv}^{2,q_1+1,2}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{2,q_1+1,3}_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\mathcal{D}_d\text{-MatDH}}_{\mathcal{B}_1}(\lambda)$.*

Proof. The algorithm $\mathcal{B}_1$ gets as input $(\mathbf{G}, g_2^{\mathbf{T}}, g_2^{\mathbf{T}\binom{\mathbf{y}}{\hat{y}}})$ as $\mathcal{D}_d$-MatDH problem instance where $\hat{y} = 0$ or $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$, $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_p^d$.

The simulator description is same as Lemma 9 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 $g_2^{k_\iota(\mathbf{Z}\binom{\mathbf{0}}{\beta}, \mathbf{0}, \mathbb{W})} \in G_2^{(d+1)}$. As $\mathcal{B}_1$ knows $\widehat{mpk}_{\text{base}}$, it chooses only one $\beta \xleftarrow{\$} \mathbb{Z}_p$ to perform the simulation. Similar to the proof of Lemma 7, we see that if $\hat{\mathbf{y}} = \mathbf{0}$, the key is a type-3 key whereas if $\hat{\mathbf{y}} \xleftarrow{\$} (\mathbb{Z}_p)^{m_2}$, the key is type-2 key.

D Security of Construction Π'_R

D.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 3.3. 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 $\overline{\mathbf{C}}^* = (\overline{\mathrm{C}}_0^*, \mathbf{C}^*, vk^*, \sigma^*)$, can however keep the same $\mathbf{C}^*$ and construct some different $\overline{\mathrm{C}}_0'$ and produce $\overline{\mathbf{C}} = (\overline{\mathrm{C}}_0', \mathbf{C}^*, vk^*, \sigma^*)$ 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.

D.2 Semi-functional Algorithms

- $\mathsf{SFEncrypt}$: Same as $\mathsf{SFEncrypt}$ in Section 3.3.1 with the exception that it runs $(vk, sk) \leftarrow \mathsf{OTS.Gen}(1^\lambda)$. Then it computes $\xi = \mathcal{H}(\mathbf{C}, vk)$ and outputs $\overline{\mathbf{C}} = (\overline{\mathrm{C}}_0, \mathbf{C}, vk, \sigma)$ where $\overline{\mathrm{C}}_0 = g_1^{(\xi \mathbf{W}_{n+1} + \mathbf{W}_{n+2})\mathbf{B}\binom{\mathbf{s}_0}{\hat{s}_0}}$ and $\sigma \leftarrow \mathsf{OTS.Sign}(sk, \overline{\mathrm{C}}_0)$.

- $\mathsf{SFAltKeyGen}$: Same as $\mathsf{SFAltKeyGen}$ in Section 3.3.1 with the exception that here the ciphertext given for decryption is of the form $\overline{\mathbf{C}} = (\overline{\mathrm{C}}_0, \mathbf{C}, vk, \sigma)$ where the commitment of $\mathbf{C}$ is computed as $\xi = \mathcal{H}(\mathbf{C}, vk)$.

D.3 Sequence of Games

Here we present indistinguishability of $\mathsf{Game}_{3,i,1}$, $\mathsf{Game}_{3,i,2}$, $\mathsf{Game}_{3,i,3}$ for $1 \leq i \leq Q$ of Table 2 in the following lemmas.

Lemma 13 ($\mathsf{Game}_{3,i-1,3}$ **to** $\mathsf{Game}_{3,i,1}$)**.** *For $i = 1, \ldots, Q$, for any efficient adversary $\mathcal{A}$ that makes at most q key queries and at most Q decryption queries, there exists a PPT algorithm $\mathcal{B}_1$ such that $|\mathsf{Adv}^{3,i-1,3}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{3,i,1}_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\mathcal{D}_d\text{-MatDH}}_{\mathcal{B}_1}(\lambda)$.*

Proof. The algorithm $\mathcal{B}_1$ gets input $(\mathbf{G}, g_2^{\mathbf{T}}, g_2^{\mathbf{T}\binom{\mathbf{y}}{\hat{y}}})$ as $\mathcal{D}_d$-MatDH problem instance where $\hat{y} = 0$ or $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$, $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_p^d$.

Setup. Same as Lemma 2. Only difference is here it chooses OTS.

Key Queries. Same as Lemma 2.

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 $\mathcal{B}_1$ computes altKey $\hat{\mathbf{K}}$ and returns $\mathsf{AltDecrypt}(\overline{\mathbf{C}}, \hat{\mathbf{K}})$ to $\mathcal{A}$. We now describe the altKey generation procedure.

- If $j > i$, it is normal altKey. As $\mathcal{B}_1$ 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. 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 $\mathbf{R} = \left(\binom{\mathbf{r}_1}{0}, \ldots, \binom{\mathbf{r}_{m_2}}{0}\right)$. It generates normal key $\mathbf{K} = (\mathrm{K}_1, \ldots, \mathrm{K}_{m_1})$ where for each $\iota \in [m_1]$, $\mathrm{K}_\iota = g_2^{k_\iota(\boldsymbol{\alpha}, \mathbf{R}, \mathbb{W})} = g_2^{\left(\begin{array}{l} b_\iota \boldsymbol{\alpha} + \sum\limits_{j \in [m_2]} b_{\iota j} \mathbf{Z}\binom{\mathbf{r}_j}{0} + \\ \sum\limits_{\substack{j \in [m_2] \\ k \in [n]}} b_{\iota jk} \mathbf{W}_k^\top \mathbf{Z}\binom{\mathbf{r}_j}{0} \end{array}\right)}$. It then computes $(\tilde{\mathrm{K}}_1, \ldots, \tilde{\mathrm{K}}_{w_1})$ where $\tilde{\mathrm{K}}_{\tilde{\iota}} = \prod\limits_{\iota \in [m_1]} (\mathrm{K}_\iota)^{\mathbf{E}_{\iota\tilde{\iota}}}$ for each $\tilde{\iota} \in [w_1]$.
  Given $\overline{\mathbf{C}} = (\overline{\mathrm{C}}_0, \mathbf{C}, vk, \sigma)$ it computes $\xi = \mathcal{H}(\mathbf{C}, vk)$. To compute the altKey, it implicitly sets $\tilde{\mathbf{Z}}^{-1}\binom{\mathbf{y}}{\hat{y}} = \binom{\mathbf{t}}{\hat{\mathrm{t}}}$. Therefore $g_2^{\mathbf{Z}\binom{\mathbf{t}}{\hat{\mathrm{t}}}} = g_2^{\tilde{\mathbf{B}}^{-\top}\mathbf{T}\binom{\mathbf{y}}{\hat{y}}}$. Then the modified key is $\hat{\mathrm{K}} = (\hat{\mathrm{K}}_0, \Phi \cdot \tilde{\mathrm{K}}_1, \tilde{\mathrm{K}}_2, \ldots, \tilde{\mathrm{K}}_{w_1})$ where $\hat{\mathrm{K}}_0 = g_2^{\mathbf{Z}\binom{\mathbf{t}}{\hat{\mathrm{t}}}}$, $\Phi = g_2^{(\xi\mathbf{W}_{n+1}^\top + \mathbf{W}_{n+2}^\top)\mathbf{Z}\binom{\mathbf{t}}{\hat{\mathrm{t}}}}$ 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. Same as Lemma 2. However, here, it runs $(vk^*, sk^*) \leftarrow \mathsf{OTS.Gen}(1^\lambda)$. Then it evaluates $\xi^* = \mathcal{H}(\mathbf{C}^*, vk^*)$ to compute $\overline{\mathbf{C}}^* = (\overline{\mathrm{C}}_0^*, \mathbf{C}^*, vk^*, \sigma^*)$ where $\overline{\mathrm{C}}_0^* = g_1^{(\xi^*\mathbf{W}_{n+1} + \mathbf{W}_{n+2})\mathbf{B}\binom{\mathbf{s}_0}{\hat{s}_0}}$ and $\sigma^* \leftarrow \mathsf{OTS.Sign}(sk^*, \overline{\mathrm{C}}_0^*)$. 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 $\mathfrak{b}'$. $\mathcal{B}_1$ outputs 1 if $\mathfrak{b}' = \mathfrak{b}$ and 0 otherwise.

Lemma 14 ($\mathsf{Game}_{3,i,1}$ to $\mathsf{Game}_{3,i,2}$). *For $i \in [Q]$, for all adversary $\mathcal{A}$ we have $|\mathsf{Adv}_{\mathcal{A}}^{3,i,1}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{3,i,2}(\lambda)| = 0$ 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 C.3) that was used to prove indistinguishability of Lemma 8 and Lemma 11. 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 $\overline{\mathbf{C}}^* = (\overline{\mathrm{C}}_0^*, \mathbf{C}^*, vk^*, \sigma^*)$ such that $\overline{\mathrm{C}}_0^* = g_1^{(\xi^*\mathbf{W}_{n+1} + \mathbf{W}_{n+2})\mathbf{B}\binom{\mathbf{s}_0}{\hat{s}_0} + \mathbf{B}\binom{\mathbf{0}}{(\xi^*\hat{\mathrm{w}}_{n+1} + \hat{\mathrm{w}}_{n+2})\hat{s}_0}}$ where $\xi^* = \mathcal{H}(\mathbf{C}^*, vk^*)$ and $\sigma^* \leftarrow \mathsf{OTS.Sign}(sk^*, \overline{\mathrm{C}}_0^*)$ 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 $\widehat{mpk}_{\text{base}}$. 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 semi-functional components of $\overline{\mathbf{C}}^*$ and $\hat{\mathbf{K}}$ are jointly identically distributed.

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

$$\left\{ g_1^{(\xi^*\mathbf{W}_{n+1}+\mathbf{W}_{n+2})\mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\ \hat{s}_0\end{smallmatrix}\right)+\mathbf{B}\left(\begin{smallmatrix}\mathbf{0}\\ (\xi^*\hat{\mathrm{w}}_{n+1}+\hat{\mathrm{w}}_{n+2})\hat{s}_0\end{smallmatrix}\right)}, g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ (\xi\hat{\mathrm{w}}_{n+1}+\hat{\mathrm{w}}_{n+2})\hat{\mathfrak{r}}\end{smallmatrix}\right)+(\xi\mathbf{W}_{n+1}^\top+\mathbf{W}_{n+2}^\top)\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ \hat{\mathfrak{r}}\end{smallmatrix}\right)} \right\}$$

and

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

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 $(\overline{\mathrm{C}}_0^*, \mathbf{C}^*, vk^*, \sigma^*) \neq (\overline{\mathrm{C}}_0, \mathbf{C}, vk, \sigma)$.

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 OTSname $(\overline{\mathrm{C}}_0, \sigma) \neq (\overline{\mathrm{C}}_0^*, \sigma^*)$.
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 $(\xi^*\hat{\mathrm{w}}_{n+1} + \hat{\mathrm{w}}_{n+2})$ and $(\xi\hat{\mathrm{w}}_{n+1} + \hat{\mathrm{w}}_{n+2})$ are pairwise independent as $\hat{\mathrm{w}}_{n+1}$ and $\hat{\mathrm{w}}_{n+2}$ are chosen uniformly at random. It implies that the semi-functional components of the ciphertext and altKey in $\mathsf{Game}_{3,i,1}$ and $\mathsf{Game}_{3,i,2}$ are identically distributed.

Lemma 15 ($\mathsf{Game}_{3,i,2}$ to $\mathsf{Game}_{3,i,3}$). *For $i = 1, \ldots, Q$, for any efficient adversary $\mathcal{A}$ that makes at most q key queries and at most Q decryption queries, there exists a PPT algorithm $\mathcal{B}_1$ such that $|\mathsf{Adv}_{\mathcal{A}}^{3,i,2}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{3,i,3}(\lambda)| \leq \mathsf{Adv}_{\mathcal{B}_1}^{\mathcal{D}_d\text{-MatDH}}(\lambda)$.*

Proof. The algorithm $\mathcal{B}_1$ gets input $(\mathbf{G}, g_2^{\mathbf{T}}, g_2^{\mathbf{T}\left(\begin{smallmatrix}\mathbf{y}\\ \hat{y}\end{smallmatrix}\right)})$ as $\mathcal{D}_d$-MatDH problem instance where $\hat{y} = 0$ or $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{T} \xleftarrow{\$} \mathcal{D}_d$, $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_p^d$.

The simulator description is same as Lemma 13 except while answering i^{th} decryption query. Here the altKey component $\hat{\mathrm{K}}_1 = \Phi \cdot \tilde{\mathrm{K}}_1$ where Φ is now multiplied by $g_2^{\mathbf{Z}\left(\begin{smallmatrix}\mathbf{0}\\ \eta u\end{smallmatrix}\right)} \in \mathbb{H}$ where $u = \sum_{\iota \in [m_1]} b_\iota \mathbf{E}_{\iota 1}$. As $\mathcal{B}_1$ knows $\widehat{mpk}_{\text{base}}$, it chooses $\eta \xleftarrow{\$} \mathbb{Z}_p$ to perform the simulation. In the similar light of Lemma 13, 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.