EFFICIENT FULLY CCA-SECURE PREDICATE ENCRYPTIONS
FROM PAIR ENCODINGS

MRIDUL NANDI
Applied Statistics Unit
Indian Statistical Institute Kolkata
West Bengal-700108, India

TAPAS PANDIT*
Department of Computer Science and Engineering
Indian Institute of Information Technology
Sri City, Chittoor
Andhra Pradesh - 517 646, India

(Communicated by Renate Scheidler)

ABSTRACT. Attrapadung (Eurocrypt 2014) proposed a generic framework for fully (adaptively) CPA-secure predicate encryption (PE) based on a new primitive, called pair encodings. Following the CCA conversions of Yamada et al. (PKC 2011, 2012) and Nandi et al. (ePrint Archive: 2015/457, AAECC 2018), one can have CCA-secure PE from CPA-secure PE if the primitive PE has either verifiability or delegation. These traditional approaches degrade the performance of the resultant CCA-secure PE scheme as compared to the primitive CPA-secure PE. As an alternative, we provide a direct fully secure CCA-construction of PE from the pair encoding scheme. This costs an extra computation of group element in encryption, three extra pairing computations and one re-randomization of key in decryption as compared to the CPA-construction of Attrapadung.

Recently, Blömer et al. (CT-RSA 2016) proposed a direct CCA-secure construction of predicate encryptions from pair encodings. Although they did not use the aforementioned traditional approaches, a sort of verifiability checking is still involved in the CCA-decryption. The number of pairing computations for this checking is nearly equal to the number of pairing computations in CPA-decryption. Therefore, the performance of our direct CCA-secure PE is far better than Blömer et al.

1. INTRODUCTION

Identity-based cryptosystem [43] was introduced to simplify certificate management process of the traditional public key cryptosystems [19, 41]. In these cryptosystems, an identity of a user is considered to be the public key. Attribute-based encryption (ABE) [26, 31, 37] is a generalization of identity-based encryption (IBE) [8, 18], a smart way to provide the access control over the secrets. In the literature, the access control that ABE implements are the boolean formulas (access structures), in the form of span programs or access trees.

2020 Mathematics Subject Classification: Primary: 94A60; Secondary: 68P25.

Key words and phrases: Pair encodings, predicate encryption, CCA-security, conversion, efficiency.

* Corresponding author: Tapas Pandit.
Similar to ABE, there are other classes of encryptions available in the literature. Some of the notable classes are (doubly-)spatial encryption ((D-)SE) [10, 27], (hierarchical) inner-product encryption ((H)IPE) [36, 39], ABE [22, 2] for circuits and functional encryption (FE) [47] for regular languages. All the aforementioned encryptions can be viewed as special cases of a unified class, called predicate encryption (PE). To design a predicate encryption scheme, first fix a tuple $(\sim, \mathcal{X}, \mathcal{Y})$, called predicate tuple, where $\mathcal{X}$ and $\mathcal{Y}$ are respectively key space and associated data space and $\sim$ is predicate or binary relation over $\mathcal{X} \times \mathcal{Y}$. In this encryption, a key is labeled with an index $x \in \mathcal{X}$, called key-index and a ciphertext is associated with another index $y \in \mathcal{Y}$, called associated data-index or simply data-index. A user who owns a key for the key-index $x$ can recover the message from a ciphertext encrypted under a data-index $y$ if a relation holds between $x$ and $y$, i.e., $x \sim y$ holds. A PE with public index hides only the message, whereas a PE with hidden index conceals both the message and the data-index. However, in this paper, we consider only the predicate encryption with public index.

The dual system methodology of Waters [45] is a well known tool for constructing adaptively secure predicate encryption scheme. But, for some predicates, e.g., regular languages, the adaptively secure predicate encryption was not known, even though their selectively-secure version was available. Therefore, for those class of predicates, the dual system technique of Waters [45] was unreachable. Recently, Attrapadung [1] introduced a new primitive, called pair encoding schemes which are implicitly contained in many predicate encryption schemes. Using the pair encodings, the author proposed a generic framework for adaptively secure predicate encryption, which captures the core technique of the dual system methodology [45]. The author [1] showed that by applying the generic approach on the pair encoding, the adaptively CPA-secure PE is possible. The conversion assumes either the perfect security or computational (doubly-selective) security of the underlying pair encoding scheme. Using this framework, the author constructed the first fully secure predicate encryption schemes for which only selectively secure schemes were known. They instantiated some surprising results, e.g., PE for regular languages, unbounded ABE for large universes, ABE with constant-size ciphertexts, etc.

Motivation: All the predicate encryption schemes of [1, 5, 48] were shown to be CPA-secure in the adaptive-predicate model. For many practical purposes, the stronger (IND-CCA) security is assumed to be mandatory for the hired encryption scheme. Using the techniques [49, 50, 33, 34], the above CPA-secure schemes can be lifted to show the CCA-security. In all these CCA conversions, a sort of index-transformation for predicate family is applied to the primitive CPA-secure PE scheme for the same family. In addition to the CPA-decryption, the CCA-decryption\(^1\) of the traditional approaches [49, 50, 33, 34] has to preform either delegation or verifiability. But the problems the above techniques suffer, are (1) increased lengths of key-indices and data-indices and (2) extra cost for performing verifiability or delegation. In the literature, most of the predicate encryption schemes are constructed using bilinear pairing groups. If the verifiability-based approach (where delegation is not known) is applied to those schemes, then checking in verifiability requires a number of pairing computations which is nearly equal to the number of pairing computations in the CPA-decryption. Altogether the techniques

---

\(^1\)In short, the decrypt algorithm of the target CCA-secure PE is denoted by CCA-decryption and the decrypt algorithm of the primitive CPA-secure PE is denoted by CPA-decryption. Similar meanings for CPA-ciphertext and CCA-ciphertext are carried throughout this paper.
degrade the performance of CCA-decryption. This leads us to ask the following questions:

*Can a direct CCA-secure PE scheme be constructed from the pair encoding scheme whose performance is very close to that of CPA-secure construction [1]?*

**Our Result.** Affirmatively, we answer the above question. That is, we provide a generic construction (see Section 3) of adaptively CCA-secure predicate encryptions from pair encodings. The high level idea is as follows: The CCA-ciphertext of our construction consists of CPA-ciphertext of [1] and a small tag (a single group element). For generating the tag, first the CPA-ciphertext is hashed using a collision resistant hash function and then the hash value is encoded by a randomness used in the CPA-ciphertext. The style of tag generation is similar to Boyen et al. [12]. Some other changes are made in decrypt algorithm to make sure that a malleable ciphertext can be detected and decrypt queries can be handled easily using dual-system proof technique [45].

It has one extra group element in ciphertext, three extra pairing computations and one re-randomization of key in decryption as compared to the CPA-decryption of [1]. For this construction, we assume two natural restrictions (see Section 2.9) on the underlying pair encoding scheme. All the underlying pair encodings and their dual [1, 5, 48] satisfy those restrictions. Therefore, we are able to achieve CCA security of all the predicate encryptions found in [1, 5, 48] at almost the same cost of CPA-construction [1].

Recently, Blömer and Liske [6] proposed a direct CCA-secure construction of predicate encryptions from pair encodings without using the traditional approaches [49, 50, 33, 34]. Their construction preserves the reduction cost of CPA-construction of [1]. Although they did not use the traditional approaches, a sort of verifiability checking is still involved before the actual CPA-decryption. The number of pairing computations for this checking is nearly equal to the number of pairing computations in CPA-decryption. Therefore, our direct CCA-secure construction of PE has far better performance than [6].

In Table 1, we provide a comparison between the performance of the decrypt algorithm of our construction and that of the construction of Blömer et al [6]. Before applying the CPA-decrypt of [1], some sort of checking is run in the decrypt algorithm of both the constructions. As mentioned earlier that two pairing computations are required before applying CPA-decrypt in our construction. In the decrypt algorithm of Blömer et al, a verifiability and other checking (to check the presence of $G_p^3$ component) are required before applying the actual CPA-decrypt and which is very costly. It varies depending upon the form of the underlying pair encoding scheme as shown in Table 1. In particular, the pairing cost for other checking is $\omega_1 + 3$, where $\omega_1$ is the size of the encoding (used in encrypt algorithm) of the pair encoding scheme. In the table, the symbols “verf” and “other” denote the numbers of pairing computations in verifiability checking and other checking respectively. The notation $\ell$ stands for either the length of string (for regular languages) or the number of rows of the span program. The symbols $m$ and $|S|$ stand for the number of transitions of a deterministic finite automaton (DFA) and the size of the attribute set $S$ respectively. The symbols $n$ and $d$ are related to the number of common variables in pair encoding scheme and the number of independent columns in the affine matrix respectively. The notations KP, CP, PES and DSE stand for key-policy, ciphertext-policy, pair encoding scheme and doubly-spatial encryption respectively. Further, the abbreviations ER, RL, LU, SU, SC and UnLU stand for...
equality relation, regular languages, large universe, small universe, short ciphertexts, and unbounded & large universe respectively. We consider the different pair encoding schemes of [1] in the table to show how better our construction performs (see the last two columns) than that of Blömer et al.

| PES of [1] | PE Scheme | Features | Additional Decryption Cost (number of pairing) | Blömer et al [6] | Our |
|------------|-----------|----------|-----------------------------------------------|------------------|-----|
| PES 1      | IBE       | ER       | 1                                             | 10              | 2   |
| PES 2      | KP-PE     | RL       | 3ℓ + 7                                         | 6               | 5ℓ + 15 |
| PES 3      | KP-ABE    | UnLU     | 4|S| + 5                                         | 2|S| + 7             | 6|S| + 12   | 2   |
| PES 4      | KP-ABE    | SC       | 8                                              | 6               | 17  |
| PES 5      | KP-ABE    | DSE      | (n + 2)|S| + 6                                         | 2|S| + 7             | (n + 4)|S| + 13 | 2   |
| PES 6      | KP-DSE    | SU       | 2|S|                                             | 3|S| + 4             | 2   |
| PES 7      | CP-PE     | SU       | 2|S|                                             | 3|S| + 4             | 2   |
| PES 8      | CP-ABE    | SU       | 2|S| + 1                                          | 2|S| + 4             | 4|S| + 5 | 2   |
| PES 9      | CP-ABE    | LU       | 2|S| + O(|B|)                                       | O(|B|) + |S| + 4 | O(|B|) + 3|S| + 5 | 2   |
| PES 10     | CP-ABE    | LU       | 2|S| + O(|B|)                                       | O(|B|) + |S| + 4 | O(|B|) + 3|S| + 5 | 2   |
| PES 11     | CP-ABE    | DSE      | n + 2                                          | d + 2           | n + d + 2 | 2   |
| PES 12     | CP-ABE    | DSE      | n + 2                                          | d + 2           | n + d + 2 | 2   |
| PES 13     | CP-ABE    | DSE      | n + 2                                          | d + 2           | n + d + 2 | 2   |
| PES 14     | CP-ABE    | DSE      | n + 2                                          | d + 2           | n + d + 2 | 2   |

Table 1. A comparison between the performance (viz., the number of pairing computations) of the decryption of our construction and that of the construction of Blömer et al.

Related Works. Fujisaki and Okamoto [21] proposed a generic efficient transformation from OW-secure PKE to CCA-secure PKE, but the security was proven in the random oracle model. The first standard model conversion which converts a CPA-secure IBE to CCA-secure PKE, was proposed by Canetti, Halevi and Katz [13]. This conversion is known as CHK-transformation and it uses one-time signature (OTS) as a supportive primitive. Subsequently, Boneh and Katz [11] improved the efficiency of CHK-transformation by replacing OTS with a weak commitment and one-time MAC. The last two conversions can be used for achieving CCA-secure IBE and HIBE [23]. For more details, the readers may consult [7].

Later, Yamada et al. [49] proposed generic conversions from CPA to CCA-secure ABE in the standard model using the properties of the underlying CPA-secure ABE, delegation and verifiability respectively. The conversion based on delegation is a generalization of [13, 7]. In the following year, Yamada et al. [50] generalized the verifiability-based conversion of [49] from ABE to PE. In the construction, the authors considered the system-index to be a single element. In the literature, there are predicates, e.g., satisfiability relation of circuits [22], general relations [37] and hierarchical inner product relations [31] whose system-indices are vectors. So, the conversion is not applicable to the corresponding PE schemes.

Recently, Nandi et al. [34] proposed a verifiability-based conversion from CPA to CCA-secure predicate encryption which is a generalization of [49] from ABE to PE. The authors claimed that the PE schemes which are realized by the conversion of [50] are also realized by their conversion. In the conversion, they considered system-index to be a vector of components as opposed to [50]. Therefore, the CPA-secure PE schemes for the aforementioned predicates can be converted to their CCA-secure variants. The work of [34] is the re-written verifiability-based part for PE of the original ePrint Archive version [33]. In the archived version, the authors also proposed a delegation-based conversion [35] which generalizes [49] from ABE to PE. All the conversion techniques in the standard model described above use a sort
of index-transformation, and delegation or verifiability property of the underlying CPA-secure PE scheme. On the other hand, we use neither of the above techniques, rather have a direct conversion from pair encodings.

**RELATED TO PAIR ENCODINGS.** In addition to fully CPA-secure construction of PE, Attrapadung [1] showed a dual conversion for pair encodings. If the source pair encoding \( P \) is perfectly secure, then the dual of \( P \), denoted by \( \mathbb{D}(P) \) is also perfectly secure encoding. Using this conversion the full security of the dual of a PE, denoted by \( \mathbb{D}(PE) \), is guaranteed if the underlying pair encoding \( P \) has the perfect security. However, there are many PE schemes for which the perfectly secure encodings were not known, so the fully secure realizations of their dual form were unsolved. Later, Attrapadung and Yamada [5] showed that the same dual conversion of [1] actually works for the computationally secure encodings. Concurrently and independently, Wee [48] proposed the notion of predicate encodings which is exactly identical to the perfectly secure pair encodings of [1]. Some of the instantiations in [48] are similar to [1], viz., the ABE for small universe with improved efficiency and doubly-spatial encryption.

A brief survey of predicate encryption is found in Appendix A.

## 2. Preliminaries

The basic notations, composite order bilinear groups, hardness assumptions, predicate family and, the syntaxes and security definitions of predicate encryption and pair encodings are provided in this section.

### 2.1. Notations

For a set \( X \), \( x \leftarrow \mathcal{R} X \) denotes that \( x \) is randomly picked from \( X \) according to the distribution \( \mathcal{R} \). Likewise, \( x \leftarrow \mathcal{U} X \) indicates \( x \) is uniformly selected from \( X \). For an algorithm \( A \) and variables \( x, y \), the notation \( x \leftarrow A(y) \) (resp. \( A(y) \rightarrow x \)) carries the meaning that when \( A \) is run on the input \( y \), it outputs \( x \). The symbol, PPT stand for probabilistic polynomial-time. For \( a, b \in \mathbb{N} \), define \([a, b] := \{i \in \mathbb{N} : a \leq i \leq b\} \) and \([b] := [1, b] \).

Throughout this paper, **bold** character denotes vector objects. For \( h \in \mathbb{Z}_N^\ell \) and \( p|N \), we define \( h \mod p := (h_1 \mod p, \ldots, h_n \mod p) \). For a vector \( x \) (resp. \( x_k \)), the \( i^{th} \) component is denoted by \( x_i \) (resp. \( x_{ki} \)). For \( x, y \in \mathbb{Z}_N^n \), we define \( <x, y> := \sum_{i=1}^n x_i \cdot y_i \).

For a matrix \( M \), the notations \( M^\top \) and \( M_{ij} \) denotes the transpose of \( M \) and entry of \( M \) at \( (i, j)^{th} \) position respectively. The notation \( M_i \) denotes the \( i^{th} \) row of the matrix \( M \). For a group \( G \) and \( n \in \mathbb{N} \), the entries from \( G^n \) are assumed to be the row vectors.

Let \( G \) be a cyclic group of order \( N \) with respect to the group operation \( \cdot \). For \( g \in G \) and \( h \in \mathbb{Z}_N^\ell \), we define \( g^h := (g^{h_1}, \ldots, g^{h_n}) \). For \( X, Y \in \mathbb{G}^n \), the notation \( X \cdot Y \) stands for component wise group operations, i.e., \( X \cdot Y := (X_1 \cdot Y_1, \ldots, X_n \cdot Y_n) \in \mathbb{G}^n \). For \( W \in \mathbb{G}^n \) and \( E \in \mathbb{Z}_N^{n \times m} \), we define \( W^E := z \in \mathbb{G}^m \), where \( z_i := W_{1}^{E_{1i}} \cdots W_{n}^{E_{ni}} \). If \( W = g^w \), for \( g \in G \) and \( w \in \mathbb{Z}_N^\ell \), then we can write \( W^E = g^{uE} \).

For a matrix \( A \in \mathbb{Z}_q^{\ell \times \ell} \), we define the linear space \( \text{Ker}(A) := \{u \in \mathbb{Z}_q^\ell \mid u^\top A = 0\} \). For \((X, x) \in \mathbb{Z}_q^{\ell \times \ell} \times \mathbb{Z}_q^\ell \), an affine space generated by \((X, x)\) is defined by \( \text{Aff}(X, x) := \{Xu + x \mid u \in \mathbb{Z}_q^\ell \} \subset \mathbb{Z}_q^\ell \). The nullity of a matrix \( A \) is defined by \( \text{Null}(A) := \text{dimension of Ker}(A^\top) \).
2.2. Composite Order Bilinear Groups. Composite order bilinear groups [9, 29] are defined to be a tuple \( \mathcal{J} := (N := p_1p_2p_3, G, G_T, e) \), where \( p_1, p_2, p_3 \) are three distinct primes and \( G \) and \( G_T \) are cyclic groups of order \( N \) and \( e : G \times G \to G_T \) is a map with the following properties:

1. (Bilinear). For all \( g, h \in G \) and \( \forall s, t \in \mathbb{Z}_p \), \( e(g^s, h^t) = e(g, h)^{st} \).
2. (Non-degenerate). There exists an element \( g \in G \) such that \( e(g, g) \) has order \( N \) in \( G_T \).
3. (Computable). There is an efficient algorithm for computing \( e(g, h) \) for all \( g, h \in G \).

Let \( G_{cbg} \) denote an algorithm which takes \( 1^\kappa \) as a security parameter and returns a description of composite order bilinear groups \( \mathcal{J} = (N = p_1p_2p_3, G, G_T, e) \). Composite order bilinear groups enjoy orthogonal property defined below.

Definition 2.1 (Orthogonal Property). Let \( G_{p_1}, G_{p_2} \) and \( G_{p_3} \) denote subgroups of \( G \) of order \( p_1, p_2 \) and \( p_3 \) respectively. The subgroups \( G_{p_1}, G_{p_2} \) and \( G_{p_3} \) are said to have orthogonal property if for all \( h_i \in G_{p_i} \) and \( h_j \in G_{p_j} \) with \( i, j \in \{1, 2, 3\} \) and \( i \neq j \), it holds that \( e(h_i, h_j) = 1 \).

Additional Notations. Let \( 1_G \) and \( 1 \) denote the identity elements of \( G \) and \( G_T \) respectively. For \( X, Y \in G^n \), we define \( e(X, Y) := \prod_{i=1}^n e(X_i, Y_i) \). For three distinct primes, \( p_1, p_2, p_3 \), a cyclic group \( G \) of order \( N = p_1p_2p_3 \), can be written as \( G = G_{p_1}G_{p_2}G_{p_3} \), where \( G_{p_i}'s \) are subgroups of \( G \) of order \( p_i \). So, each element \( X \in G \) can be expressed as \( X = X_1X_2X_3 \), where \( X_i \in G_{p_i} \). For \( X, Y \in G^n \), let \( Y|_{G_{p_i}} \) denote \( (Y_1|_{G_{p_1}}, \ldots, Y_n|_{G_{p_n}}) \). Let \( g_T \) stand for the element \( e(g, g) \), where \( g \in G_{p_i} \).

2.3. Hardness Assumptions in Composite Order Bilinear Groups. We describe here three Decisional SubGroup (DSG) assumptions [31] for 3 primes, DSG1, DSG2 and DSS3 in composite order bilinear groups. Let \( \mathcal{J} := (N = p_1p_2p_3, G, G_T, e) \leftarrow G_{cbg}(1^\kappa) \) be the common parameters for each assumptions. In the following, we define instance for each assumption.

- **DSG1.** Let \( g \leftarrow G_{p_1}; Z_3 \leftarrow G_{p_3}; T_0 \leftarrow G_{p_1}; T_1 \leftarrow G_{p_1}p_2; \) Define \( D := (\mathcal{J}, g, Z_3) \).
- **DSG2.** Let \( g, Z_1 \leftarrow G_{p_1}; Z_2, W_2 \leftarrow G_{p_2}; Z_3 \leftarrow G_{p_3}; T_0 \leftarrow G_{p_1}p_3; T_1 \leftarrow G \). Then define \( D := (\mathcal{J}, g, Z_1Z_2, W_2W_3, Z_3) \).
- **DSG3.** Let \( \alpha, s \leftarrow \mathbb{Z}_N; g \leftarrow G_{p_1}; W_2, Y_2, g_2 \leftarrow G_{p_2}; Z_3 \leftarrow G_{p_3}; T_0 := e(g, g)^{\alpha s}; T_1 \leftarrow G_{T';} \) Define \( D := (\mathcal{J}, g, g_2Y_2, g_2W_2, g_2, Z_3) \).

The advantage of an algorithm \( \mathcal{A} \) in breaking DSGi, for \( i = 1, 2, 3 \) is defined by

\[
\text{Adv}_{\mathcal{A}}^{\text{DSGi}}(\kappa) = |\text{Pr}[\mathcal{A}(D, T_0) = 1] - \text{Pr}[\mathcal{A}(D, T_1) = 1]|.
\]

We say that the DSGi assumption holds in \( \mathcal{J} \) if for every PPT algorithm \( \mathcal{A} \), the advantage \( \text{Adv}_{\mathcal{A}}^{\text{DSGi}}(\kappa) \) is negligible in security parameter \( \kappa \).

2.4. Predicate Family. To define a predicate-based cryptosystem, we have to define predicate family. The predicate family is defined for an index set \( \Lambda \). For most of the predicate families, the index sets are considered to be subsets of \( \{j : j \in \mathbb{N} \text{ and } i \in \mathbb{N}\} \). The following definition of predicate family is adopted from [6, 1].
Definition 2.2 (Predicate Family). For an arbitrary index set \( \Lambda \), we define predicate family to be \( \sim := \{ \sim_j \}_{j \in \Lambda} \), where \( \sim_j : \mathcal{X}_j \times \mathcal{Y}_j \to \{0, 1\} \) is an indicator function, and \( \mathcal{X}_j \) and \( \mathcal{Y}_j \) are respectively called key space and associative data space.

The function \( \sim_j \) is also called predicate or binary relation over \( \mathcal{X}_j \times \mathcal{Y}_j \). For \( (x, y) \in \mathcal{X}_j \times \mathcal{Y}_j \), we write \( x \sim_j y \) if \( \sim_j (x, y) = 1 \) else \( x \not\sim_j y \). For a predicate family, the corresponding index set \( \Lambda \) is called system-index space. A member \( j \) of the index space \( \Lambda \) is called index for system parameter or simply system-index. To design a predicate-based scheme for some predicate family, first a system-index \( j \) is fixed for that family. Then, this index will define a predicate tuple \( (\sim_j, \mathcal{X}_j, \mathcal{Y}_j) \) for the corresponding predicate-based scheme. For example, the system-indices for predicate families, regular languages, circuits, access structures, inner product and doubly-spatial relation are respectively alphabet, maximum depth and number variables for circuits, attribute universe or size of the attribute universe, length of vectors and dimension of affine space.

In the following, we describe some of the predicates widely used in practice. Note that for most of the relations described below, the system-indices are not given explicitly as it will be understood from the context.

Equality relation. Let \( \mathcal{X} = \mathcal{Y} = \{0, 1\}^* \). For \( x, y \in \{0, 1\}^* \), we define \( x \sim y \) if and only if \( x = y \). The well known predicate encryption for the equality relation is called identity-based encryption (IBE).

Inner product relation. Let \( \mathcal{X} = \mathcal{Y} = \mathbb{Z}_q^\ell \). For \( x = (x_1, \ldots, x_k) \in \mathcal{X} \) and \( y = (y_1, \ldots, y_k) \in \mathcal{Y} \), we define \( x \sim y \) if and only if \( \langle x, y \rangle = 0 \). This relation is called zero inner product relation. Similarly, a non-zero inner product relation is defined by \( x \sim y \) if and only if \( \langle x, y \rangle \neq 0 \). The corresponding encryption schemes are known as inner-product encryption (IPE).

(Doubly)-spatial relation. \( \mathcal{X} = \mathcal{Y} := \{\text{Aff}(A, a) : (A, a) \in \mathbb{Z}_q^{\ell \times k} \times \mathbb{Z}_q^{k \times \ell}, 0 \leq k \leq \ell\} \).

For \( x \in \mathcal{X} \) and \( y \in \mathcal{Y} \), doubly-spatial relation is defined by \( x \sim_{ds} y \) if and only if \( y \cap x \neq \emptyset \). For spatial relation, we restrict \( \mathcal{Y} \) to be \( \mathbb{Z}_q^\ell \). In [17], the doubly-spatial relation was defined over \( \mathcal{X} \times \mathcal{Y} \), where \( \mathcal{X} := \{\text{Aff(X)} : X \in \mathbb{Z}_q^{\ell \times k}, 0 \leq \ell \} \) and \( \mathcal{Y} := \{\text{Aff}(A, a) : (A, a) \in \mathbb{Z}_q^{\ell \times \ell} \times \mathbb{Z}_q^{k \times \ell}, 0 \leq k \leq \ell\} \). The predicate encryption using the (doubly)-spatial relation is called (doubly)-spatial encryption ((D)SE). The authors in [17] showed that predicate encryption for doubly-spatial relation defined later generalizes the predicate encryption for the former defined doubly-spatial relation.

Access structure based relation. Let \( \mathcal{U} \) be a universe of attributes. Define \( \mathcal{X} = 2^\ell \) and \( \mathcal{Y} \) be the set of all access structures over \( \mathcal{U} \). For \( A \in \mathcal{X} \) and \( \Gamma \in \mathcal{Y} \), we define a binary relation \( A \sim \Gamma \) if and only if \( A \in \Gamma \). The encryption scheme realizing this relation is called attribute-based encryption (ABE) for access structures.

Policy over doubly-spatial relation. We have defined access structure based relation above through the equality relation over universe of attributes. Here we define a new access structure based relation of [1], called policy over doubly-spatial relation using the doubly-spatial relation over universe of affine subspaces. This predicate generalizes the former access structure based relation. Let \( \ell \) be a system-index for this new access structure based relation. We define \( \mathcal{U} := \{\text{Aff}(A, a) : (A, a) \in \mathbb{Z}_q^{\ell \times k} \times \mathbb{Z}_q^{k \times \ell}, 0 \leq k \leq \ell\} \). Let \( \mathcal{X} := 2^{\ell r} \) and \( \mathcal{Y} \) be the set of all policies of the form \( (M, \rho) \), where \( M \in \mathbb{Z}_q^{\ell \times r} \) and \( \rho : [d] \to \mathcal{U} \) is a row labeling function. For \( S := \{Y_1, \ldots, Y_t\} \in \mathcal{X} \) and \( \Lambda := (M, \rho) \in \mathcal{Y} \), we define
For notational simplicity, we omit $j$ predicate. We therefore reserve the first entry of Definition 2.3 (Symmetric predicate). This describes some domain, for example, the domain of IBE is all the predicates of $[1, 48]$ are domain-transferable. Since, all other predicates are defined through the equality predicate, all the predicates of $[1, 48]$ are domain-transferable.

Acceptance relation in regular language. A deterministic finite automaton $M$ is defined to be a quintuple $(Q, \Sigma, \delta, q_0, F)$, where $Q$ is a finite set of states, $\Sigma$ is a finite set of symbols, called alphabet, $q_0 \in Q$ is called the start state, $F \subseteq Q$ is called the set of final states and $\delta : Q \times \Sigma \rightarrow Q$ is called transition function. The language, also called regular language, recognized by a deterministic finite automaton (DFA) $M$ is defined as

$$\mathcal{L}(M) = \{ \sigma_1 \sigma_2 \cdots \sigma_n \in \Sigma^* : \delta(\sigma_1, \sigma_2) \cdots \delta(\sigma_{n-1}, \sigma_n) = q_f \}.$$ 

Let $\text{Tr}$ denote the set of all transitions $(q_x, q_y, \sigma) \in Q \times Q \times \Sigma$ with the understanding that $\delta(q_x, \sigma) = q_y$. If we identify the $\delta$ by $\text{Tr}$, then a DFA $M$ can always be represented by $(Q, \Sigma, \text{Tr}, q_0, F)$. Let $\Sigma$ be an alphabet, $\mathcal{X} := \Sigma^*$ and $\mathcal{Y}$ be the set of all DFAs with the same alphabet $\Sigma$. For $w \in \mathcal{X}$ and $M \in \mathcal{Y}$, we define a binary relation $w \sim M$ if $w \in \mathcal{L}(M)$. We also call this relation as DFA-based relation. The corresponding encryption scheme is known as functional encryption (FE) [47] for regular languages.

**Definition 2.3** (Symmetric predicate). A relation defined over $\mathcal{X} \times \mathcal{Y}$ is called symmetric if $\mathcal{X} = \mathcal{Y}$ and $x \sim y \iff y \sim x$ for all $x, y \in \mathcal{X}$, otherwise it is called asymmetric.

For an asymmetric relation, we can define its dual relation as follows.

**Definition 2.4** (Dual predicate). For a predicate tuple $(\sim, \mathcal{X}, \mathcal{Y})$, its dual predicate tuple $(\sim, \mathcal{X}', \mathcal{Y}')$ is defined by $\mathcal{X}' := \mathcal{Y}$, $\mathcal{Y}' := \mathcal{X}$ and for $(x, y) \in \mathcal{X} \times \mathcal{Y}$, $x \sim y$ holds if and only if $y \sim x$ holds. The predicate $\sim$ is called dual predicate of $\sim$.

**Remark 1.** In this paper, we consider predicate encryption for all the relations described above and their dual (for asymmetric relations). If the underlying predicate or relation of PE is not clearly stated, we assume that the PE stand for one of the aforementioned relations.

Here we are interested to design an adaptively CCA-secure predicate encryption over composite order bilinear groups (CBG) and let $N$ be the order of the groups. This $N$ describes some domain, for example, the domain of IBE is $\mathbb{Z}_N$ with equality predicate. We therefore reserve the first entry of $j$ to be $N$ as described in [1]. For notational simplicity, we omit $j$ and write $(\sim_N, \mathcal{X}_N, \mathcal{Y}_N)$ or simply $(\sim, \mathcal{X}, \mathcal{Y})$ depending upon requirement.

**Definition 2.5.** (Domain-transferable [1]). We say that $\sim$ is domain-transferable if for $p$ dividing $N$, the projection map $f_1 : \mathcal{X}_N \rightarrow \mathcal{X}_p$ and $f_2 : \mathcal{Y}_N \rightarrow \mathcal{Y}_p$ such that for all $(x, y) \in \mathcal{X}_N \times \mathcal{Y}_N$, we have:

- (Completeness). If $x \sim_N y$ then $f_1(x) \sim_p f_2(y)$.
- (Soundness). (1) If $x \not\sim_N y$, then $f_1(x) \not\sim_p f_2(y)$ or (2) there exists an algorithm which takes $(x, y)$ as input, where (1) does not hold, outputs a non-trivial factor $F$ such that $p|F|N$.

**Remark 2.** Attrapadung [1] showed that the equality predicate (for IBE) is domain-transferable. Since, all other predicates are defined through the equality predicate, all the predicates of [1, 48] are domain-transferable.
2.5. Predicate Encryption. A predicate encryption (PE) scheme for a predicate family \( \sim \) consists of four PPT algorithms - Setup, KeyGen, Enc and Decrypt.

- **Setup**: It takes a security parameter \( \kappa \) and a system-index \( j \) as input, outputs public parameters \( \mathcal{PP} \) and master secret key \( \text{MSK} \).
- **KeyGen**: It takes as input \( \mathcal{PP} \), \( \text{MSK} \) and a key-index \( x \in \mathcal{X} \) and outputs a secret key \( \text{SK}_x \) corresponding to \( x \).
- **Enc**: It takes \( \mathcal{PP} \), a message \( m \in \mathcal{M} \) and an associated data-index \( y \in \mathcal{Y} \) and returns a ciphertext \( C \), which implicitly contains \( y \).
- **Decrypt**: It takes as input \( \mathcal{PP} \), a ciphertext \( C \) and a key \( \text{SK}_x \). It returns a value from \( \mathcal{M} \cup \{ \bot \} \).

Correctness. For all \( (\mathcal{PP}, \text{MSK}) \leftarrow \text{Setup}(1^\kappa, j) \), all \( y \in \mathcal{Y} \), all \( x \in \mathcal{X} \), \( \text{SK}_x \leftarrow \text{KeyGen}(\mathcal{PP}, \text{MSK}, x) \) and for all messages \( m \in \mathcal{M} \), it is required that

\[
\text{Decrypt}(\mathcal{PP}, \text{Enc}(\mathcal{PP}, m, y), \text{SK}_x) = m \quad \text{(resp. } \bot) \quad \text{if } x \sim y \quad \text{(resp. } x \not\sim y) \).
\]

2.6. Security of Predicate Encryption.

**Definition 2.6** (Adaptive-Predicate IND-CCA Security). A PE scheme is said to be IND-CCA secure in adaptive-predicate model (or simply AP-IND-CCA secure or fully CCA-secure or adaptively CCA-secure) if for all PPT algorithms \( \mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2) \), the advantage

\[
\text{Adv}_{\mathcal{A}, \mathcal{PE}}^{\text{AP-IND-CCA}}(\kappa) := \Pr[b = b' \wedge \text{NRn}] - \frac{1}{2}
\]

in \( \text{Exp}_{\mathcal{A}, \mathcal{PE}}^{\text{AP-IND-CCA}}(\kappa) \) defined in Figure 1 is negligible function in security parameter \( \kappa \), where \( \mathcal{A} \) is provided access to key-gen oracle \( \mathcal{O}_K \) and decrypt oracle \( \mathcal{O}_D \) (described below) and NRn is a natural restriction that \((C^*, x)\) with \( x \sim y^* \) was never queried to \( \mathcal{O}_D \) and for each key-index \( x \) queried to \( \mathcal{O}_K \), it holds that \( x \not\sim y^* \).

- The challenger maintains a log \( \mathcal{L} \) for storing the pairs of the form \( (x, \text{SK}_x) \) and initially sets \( \mathcal{L} \leftarrow \emptyset \).
- **KeyGen oracle** \( \mathcal{O}_K \): Given a key-index \( x \), it first searches \( x \) in the log \( \mathcal{L} \). If \( (x, \text{SK}_x) \) is found in \( \mathcal{L} \), oracle returns \( \text{SK}_x \). Otherwise it runs \( \text{SK}_x \leftarrow \text{KeyGen}(\mathcal{PP}, \text{MSK}, x) \), adds \( (x, \text{SK}_x) \) to \( \mathcal{L} \) and returns \( \text{SK}_x \).
- **Decrypt oracle** \( \mathcal{O}_D \): Given \( (C, x) \), it first searches \( x \) in the log \( \mathcal{L} \). If \( (x, \text{SK}_x) \) is found in \( \mathcal{L} \), then returns \( \text{Decrypt}(\mathcal{PP}, C, \text{SK}_x) \). Otherwise, it runs \( \text{SK}_x \leftarrow \text{KeyGen}(\mathcal{PP}, \text{MSK}, x) \), adds \( (x, \text{SK}_x) \) to \( \mathcal{L} \) and returns \( \text{Decrypt}(\mathcal{PP}, C, \text{SK}_x) \).

![Figure 1](image-url)

**Figure 1.** Experiment for confidentiality (adaptive-predicate IND-CCA Security)

A weaker notion of security can be defined similarly as above except, \( \mathcal{A} \) is not allowed to access to \( \mathcal{O}_D \) oracle. It is called IND-CPA security in both adaptive-predicate and selective predicate models.
2.7. Pair encoding scheme. A Pair Encoding Scheme [1] $P$ for a predicate family, $\sim$ consists of four deterministic algorithms, Param, Enc1, Enc2 and Pair.

- **Param($j$) $\rightarrow$ $n \in \mathbb{N}$.** $n$ describes the number of common variables involved in Enc1 and Enc2. Let $h := (h_1, \ldots, h_n) \in \mathbb{Z}_N^n$ denotes the common variables in Enc1 and Enc2.

- **Enc1($x \in \mathcal{X}, N$) $\rightarrow$ ($k_x := (k_{1}, \ldots, k_{m_x}), m_2$),** where $k_i$’s for $i \in [m_1]$ are polynomial over $\mathbb{Z}_N$ and $m_2 \in \mathbb{N}$ specifies the number of its own variables.

- **Enc2($y \in \mathcal{Y}, N$) $\rightarrow$ ($c_y := (c_{1}, \ldots, c_{\omega_2}), \omega_2$),** where $c_i$’s for $i \in [\omega_1]$ are polynomial over $\mathbb{Z}_N$ and $\omega_2 \in \mathbb{N}$ specifies the number of its own variables.

- **Pair($x, y, N$) $\rightarrow$ $E \in \mathbb{Z}_N^{m_1 \times \omega_1}$.**

Correctness:

1. For all $N \in \mathbb{N}$, ($k_x, m_2$) $\leftarrow$ Enc1($x, N$), ($c_y, \omega_2$) $\leftarrow$ Enc2($y, N$), and $E \leftarrow$ Pair($x, y, N$), we have $k_x(\alpha, r, h)E_{c_y}^j(s, h) = \alpha s_0$ if $x \sim y$.

2. For $p|N$, if ($k_x, m_2$) $\leftarrow$ Enc1($x, N$) and ($k_y, m_2$) $\leftarrow$ Enc1($x, p$), then we require that $k'_x = k_x \mod p$. Similar type of condition is required for Enc2.

Properties of pair encoding scheme. We define two properties of pair encoding scheme as follows

- **(Param-Vanishing):** $k(\alpha, 0, h) = k(\alpha, 0, 0)$.

- **(Linearity):**

\[
\begin{align*}
    k(\alpha_1, r_1, h) + k(\alpha_2, r_2, h) &= k(\alpha_1 + \alpha_2, r_1 + r_2, h) \\
    c(s_1, h) + c(s_2, h) &= c(s_1 + s_2, h).
\end{align*}
\]

2.8. Security of pair encoding scheme. We consider two forms of security, viz., perfect security and computational security as defined in [1].

- **Perfect Security:** A pair encoding scheme is said to be **perfectly master-key hiding (PMKH)** if for $N \in \mathbb{N}$, $x \not\sim_N y$, $n \leftarrow$ Param($j$), ($k_x, m_2$) $\leftarrow$ Enc1($x, N$) and ($c_y, \omega_2$) $\leftarrow$ Enc2($y, N$), the following two distributions are identical:

\[
\{c_y(s, h), k_x(\alpha, r, h)\} \text{ and } \{c_y(s, h), k_x(0, r, h)\}
\]

where the random coins of the distributions are $\alpha \leftarrow \mathbb{Z}_N$, $h \leftarrow \mathbb{Z}_N^n$, $s \leftarrow \mathbb{Z}_{N^{\omega_2+1}}$ and $r \leftarrow \mathbb{Z}_{N^2}^{m_2}$. 
- Computational Security: Here we consider two types of computational security, viz., selectively master-key hiding (SMH) and co-selectively master-key hiding (CMH). A pair encoding scheme is said to have \( G \) security for \( G \in \{\text{SMH}, \text{CMH}\} \) if for \( b \leftarrow U \{0,1\} \), all PPT adversary \( \mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2) \), the advantage \( \text{Adv}^G_{\mathcal{A},b}(\kappa) := |\Pr[\text{Exp}^G_{\mathcal{A},0}(\kappa) = 1] - \Pr[\text{Exp}^G_{\mathcal{A},1}(\kappa) = 1]| \) in the experiment \( \text{Exp}^G_{\mathcal{A},b}(\kappa) \) defined below is negligible function in security parameter \( \kappa \):

\[
\text{Exp}^G_{\mathcal{A},b}(\kappa) := \begin{cases} 
(N := p_1p_2p_3, G, G_T, e) \leftarrow G_{\text{cgb}}(1^\kappa); \\
(g, g_2, g_3) \leftarrow U G_{p_1} \times G_{p_2} \times G_{p_3}; \\
\alpha \leftarrow U \mathbb{N}_N; \ n \leftarrow \text{Param}(j); \ h \leftarrow U \mathbb{Z}_N^2; \\
st \leftarrow \mathcal{A}_1^{G,\alpha,b,h}(g, g_2, g_3); \\
b' \leftarrow \mathcal{A}_2^{G,\alpha,b,h}(st)
\end{cases}
\]

where \( \mathcal{A} \) is provided the access to two oracles, \( \mathcal{O}^1_{G,b,\alpha,h}(\cdot) \) and \( \mathcal{O}^2_{G,b,\alpha,h}(\cdot) \) defined below:

- For Selective Security: \( \mathcal{O}^1 \) is allowed only once, while \( \mathcal{O}^2 \) is allowed to query polynomially many times
  * \( \mathcal{O}^1_{\text{SMH},b,\alpha,h}(y^*): \) Run \((c_{y^*}, \omega_2) \leftarrow \text{Enc2}(y^*, p_2)\), pick \( s \leftarrow U \mathbb{Z}_{2^\omega_2+1}^N \) and return \( C_{y^*} := g_2^{c_{y^*}(s, h)} \).
  * \( \mathcal{O}^2_{\text{SMH},b,\alpha,h}(x): \) If \( x \sim_{p_2} y^* \), return \( \perp \). Run \((k_x, m_2) \leftarrow \text{Enc1}(x, p_2)\), pick \( r \leftarrow U \mathbb{Z}_{N^2_{m_2}} \) and return

\[
K_x := \begin{cases} 
g_2^{k_2(0, r, h)} & \text{if } b = 0 \\
g_2^{k_2(\alpha, r, h)} & \text{if } b = 1.
\end{cases}
\]

- For Co-selective Security: Both the oracles, \( \mathcal{O}^1 \) and \( \mathcal{O}^2 \) are allowed to query only once.
  * \( \mathcal{O}^1_{\text{CMH},b,\alpha,h}(x^*): \) Run \((k_{x^*}, m_2) \leftarrow \text{Enc1}(x^*, p_2)\), pick \( r \leftarrow U \mathbb{Z}_{N^2_{m_2}} \) and then return

\[
K_{x^*} := \begin{cases} 
g_2^{k_{2}(0, r, h)} & \text{if } b = 0 \\
g_2^{k_{2}(\alpha, r, h)} & \text{if } b = 1.
\end{cases}
\]
  * \( \mathcal{O}^2_{\text{CMH},b,\alpha,h}(y): \) If \( x^* \sim_{p_2} y \), return \( \perp \). Run \((c_y, \omega_2) \leftarrow \text{Enc2}(y, p_2)\), pick \( s \leftarrow U \mathbb{Z}_{2^\omega+1}^N \) and then return \( C_y := g_2^{c_y(s, h)} \).

Remark 3. In the above definition of computational security, if the oracles, \( \mathcal{O}^1 \) and \( \mathcal{O}^2 \) are allowed to access respectively \( t_1 \) and \( t_2 \) times, then SMH (resp. CMH)-security, will be referred as \( (t_1, t_2)-\text{SMH} \) (resp. \( (t_1, t_2)-\text{CMH} \)) security. What considered in [1], are \((1, poly)-\text{SMH} \) and \((1, 1)-\text{CMH} \) security respectively for selectively and co-selectively master-key hiding. It is clear from the definitions of PMH and CMH-security that the PMH-security of a pair encoding scheme implies the CMH-security.

2.9. Natural Requirements on Pair Encodings. Below, we define two restrictions on pair encoding scheme which are required for correctness and security proof of the proposed construction in Section 3.1.
Conditions 2.7 (Sufficient). We put the following conditions on the pair encodings. To the best of our knowledge, most of the pair encoding schemes satisfy these conditions.

1. \( c_i(s, h) = s_0 \) for some \( i \in [\omega_1] \).
2. For \( (x, y) \in X \times Y \) with \( x \sim y \), \( (k_x, m_2) \leftarrow \text{Enc1}(x, N) \) and \( E \leftarrow \text{Pair}(x, y, N) \), we require that \( k_x(\alpha, 0, 0)E := (*, 0, \ldots, 0) \in \mathbb{Z}_N^e \), where * is any entry from \( \mathbb{Z}_N \).

For better understanding the above conditions, we work out on the pair encoding scheme for unbounded KP-ABE with large universes (Scheme 4 of [1]) illustrated in Figure 2. We show that this pair encoding satisfies Conditions 2.7. The condition (1) is so obvious. For verifying the condition (2), we first notice that \( k_x(\alpha, 0, 0) = (\alpha, 0, \ldots, 0) \). Hence, we have to show that \( E_{ij} = 0 \) for \( j \in [2, \omega_1] \). From the correctness of the scheme, we find that the monomials containing \( k \) appear in the correctness are exactly \( k_1 \), so the first row of the matrix \( E \) must be \((1, 0)\). Hence, we are done.

2.10. Dual Conversion of Pair Encodings. We illustrate the dual conversion technique [1, 5] for converting a pair encoding for \( \sim \) to another pair encoding for its dual predicate (Definition 2.4) \( \bar{\sim} \).

Let \( P \) be a given pair encoding scheme for the predicate \( \sim \). A pair encoding scheme \( \mathbb{D}[P] \) for the predicate \( \bar{\sim} \) is defined as follows: For \( (n, h) \leftarrow \text{Param} \), we define \( \text{Param} := (n + 1, \bar{h}) \), where \( \bar{h} := (h, \phi) \) and \( \phi \) is a new variable.

- \( \text{Enc1}(x, N) \): It runs \( (c'_y(s', \bar{h}, \omega_2) \leftarrow \text{Enc2}(x, N) \), where \( s' := (s_0', \ldots, s'_2) \). Then sets \( r := s' \) and \( k_x(\alpha, r, \bar{h}) := (c'_y(s', h), \alpha + \phi \cdot s') \). Finally, it outputs \( (k_x(\alpha, r, \bar{h}), \omega_2) \), where \( \alpha \) is a new variable.
- \( \text{Enc2}(y, N) \): Runs \( (k'_y(\alpha', r', m_2) \leftarrow \text{Enc1}(y, N) \). Then sets \( s := (s_0, r') \) and \( c_y(s, \bar{h}) := (k'_y(\phi \cdot s_0, s, \bar{h}), s_0) \), and returns \( (c_y(s, \bar{h}), m_2) \), where \( s_0 \) is a new variable.

Figure 2. A brief description of the pair encoding scheme (Scheme 4 of [1]) used in the construction of unbounded KP-ABE with large universes.
The correctness is verified as follows: If \( x \sim y \), then \( y \sim x \), so from the correctness of \( P \) we have

\[
k'_y(\alpha', r', h)E'c_x'(s', h) = \alpha' s'_0 = (\phi \cdot s_0)s'_0.
\]

Then using the additional components, we have \((\alpha + \phi \cdot s'_0)(s_0) - (\phi \cdot s_0)s'_0 = \alpha s_0\).

**Proposition 1.** ([1]) If a pair encoding scheme \( P \) for \( \sim \) is perfectly master-key hiding, then the pair encoding scheme \( D(P) \) for \( \sim \) is also perfectly master-key hiding.

**Proposition 2.** ([5]) If a pair encoding scheme \( P \) for \( \sim \) is normal and \((1, 1)\)-co-selectively master-key hiding, then the pair encoding scheme \( D(P) \) for \( \sim \) is \((1, 1)\)-selectively master-key hiding.

**Proposition 3.** ([5]) If a pair encoding scheme \( P \) for \( \sim \) is normal and \((1, 1)\)-co-selectively master-key hiding, then the pair encoding scheme \( D(P) \) for \( \sim \) is \((1, 1)\)-co-selectively master-key hiding.

**Observation 2.8.** We first note that the pair encoding scheme, \( D(P) \) satisfies the condition (1) of Conditions 2.7 due to newly added variable \( s_0 \). Let us examine condition (2). W.l.o.g, we set \( c_{y,1} = s_0 \) and \( k_{x,1} = \alpha + \phi \cdot s'_0 \). The correctness of \( D(P) \) says that \( k_x(\alpha, r, h)E(c_y(s, h)) = k_{x,1} \cdot c_{y,1} - k'_y(\alpha', r', h)E'(c'_x(s', h)) = \alpha s_0 \). If \( E' \) has dimension \((m_1 \times \omega'_1)\), then the dimension of \( E \) is \((m_1 \times \omega_1)\), where \( m_1 = \omega'_1 + 1 \) and \( \omega_1 = m_1 + 1 \). Hence, the matrix, \( E \) has the following form:

\[
E_{ij} := \begin{cases} 
1 & \text{if } i = 1, j = 1 \\
0 & \text{if } i = 1, j \in [2, \omega_1] \\
0 & \text{if } i \in [2, m_1], j = 1 \\
-E'_{(j-1)(i-1)} & \text{if } i \in [2, m_1], j \in [2, \omega_1].
\end{cases}
\]

Therefore, it is straightforward to check that the dual pair encoding scheme \( D(P) \) satisfies the condition (2) of Conditions 2.7. So far, we check that dual of all the pair encoding schemes [1, 5, 48] satisfy Conditions 2.7. Therefore, all the pair encoding schemes of [1, 5, 48] and their dual satisfy Conditions 2.7 and, have either computational security (CMH and SMH) or PMH security.

3. **Direct CCA-secure predicate encryption**

Before describing our CCA-secure construction more formally, we first discuss outline of our construction as follows.

**Outline of Our Construction.** One major concern in designing a CCA-secure PE is that new ciphertext (mostly well-formed or ill-formed but up to a certain extent) must not be created from a given ciphertext. In the traditional approaches [49, 50, 33, 34], the aforementioned concern is handled by using a strongly unforgeable OTS scheme. However, in this direct CCA-secure construction, we neither follow the traditional approaches nor use OTS scheme. Rather, we extend CPA-construction of [1] to CCA-construction by considering an alternative of OTS. We first recall that a ciphertext and key in CPA-construction of [1] are of the forms \( C_{\text{cga}} := (y, C_y := g_{c_y(s, h)}, C_{\text{INT}} := m \cdot g_{c_y^r}^m) \) and \( SK_x := (x, K_x := g^{k_x(\alpha, r, h)}, R_3) \) with \( R_3 \in \mathbb{G}_m \). The CPA-decryption returns \( C_{\text{INT}}/e(K_x^{E_y}, C_y) \), where \( E \leftarrow \text{Pair}(x, y) \).

We note that a generator of \( \mathbb{G}_{p_3} \) is given as a part of the public parameters in the CPA-construction. The construction [1] cannot be CCA-secure as \( C_y \) part of \( C_{\text{cga}} \) can be made ill-format by composing with the elements of \( \mathbb{G}_{p_3} \). This ill-format can be recognized if each component of \( C_y \) is checked using extra pairing computations.
construction is described as follows: For CCA-construction, we add a natural encoding to construct CCA-secure version of [1]. The intuitive idea of our construction is described as follows: For CCA-construction, we add $g^{θ_1}, g^{θ_2}$ and a collision-resistant hash function $H : \{0, 1\}^* → \mathbb{Z}_N$ to the public parameters of CPA-construction. A CPA-ciphertext $C_{cpa}$ is first hashed to $h := H(C_{cpa})$ and then $h$ is encoded (locked) to $C_0 := g^{s_0(h, h_0)}$ using $s_0, g^{θ_1}$ and $g^{θ_2}$. Now, a CCA-ciphertext of our proposed construction will be of the form $CT := (C_{cpa}, C_0)$. The hash value $h$ takes care of preserving the well-formedness of $C_{cpa}$, whereas the new component $C_0$ keeps track of the actual value $h = H(C_{cpa})$. By natural restriction (condition (1) of Conditions 2.7) on pair encoding scheme, we have that the first component of $g^{e_v(s, h)}$ is $g^{s_0}$. Without explicit knowledge of $s_0$, the component $C_0$ is computationally hard to compute from the available objects $h, g^{s_0}, g^{θ_1}$ and $g^{θ_2}$. In other word, $C_0$ can be thought as a natural replacement for OTS in the traditional approaches, where $s_0$ plays a role like singing key in OTS scheme. If $s_0$ is not compromised and $C_{cpa}$ is changed, then we can recognize this change by checking the following equation using $g, C_0, h = H(C_{cpa}), g^{s_0}, g^{θ_1}$ and $g^{θ_2}$:

$e(g, C_0) = e(g^{θ_1 h + θ_2}, g^{s_0})$

Now, the aforementioned concern boils down to checking the well-formedness (or ill-format, but up to certain extent) of $C_0$, viz., to check whether $C_0$ contains the element of $\mathbb{G}_{p_3}$ or not. If we replace equation (1) by the following equation, then the later checking can be done automatically, where $R ← \mathcal{U} \mathbb{G}_{p_3}$:

$e(g · R, C_0) = e(g^{θ_1 h + θ_2}, g^{s_0})$

Given a CCA-ciphertext $CT$, first the well-formedness (or ill-format, but up to certain extent) of $CT$ is checked as discussed above. Note that for this checking, we require two pairing computations. Then the actual CPA-decryption (but, slightly different way) is run to recover the underlying message.

Another major concern is to show the adaptive security of the proposed construction. We extend the dual system proof style of [1] in novel way. In this approach, security is proven by applying the hybrid arguments over a polynomial number of hybrid games. In addition to answering the various queries of CPA-construction, a simulator has to handle a polynomial number of decrypt queries made by adversary. To smoothen the hybrid arguments, the actual CPA-decryption is slightly modified as shown below: Decrypter first re-randomizes the key $SK_x$ and then creates an alternative key $SK_x^M$ using the key $SK_x$ as follows: $SK_x^M := (K_0, \Psi, \Phi^M) ∈ \mathbb{G}^{ω_1 + 1}$, where $K_0 := g^{-τ} · R_0$, $\Psi := g^θ$ with $θ := (τ(θ_1 h + θ_2), 0, \ldots, 0) ∈ Z_N^{ω_1}$, $τ ← \mathcal{U} Z_N$, $R_0 ← \mathbb{G}_{p_3}$ and $\Phi ← \text{Pair}(x, y)$. Then, it recovers the underlying message as

$2$If a ciphertext is neither well-formed nor ill-formed but up to certain extent, decrypt algorithm will return $\perp$.

$3$This technique was first used by Boyen et al. [12] to construct CCA-secure PKE and HIBE schemes. Later, this technique was used in many other cryptographic constructions. Recently, Blömer and Liske [6] used the same technique to construct adaptively CCA-secure PE scheme from encoding scheme.
CCA-secure construction from pair encodings. For doing this, an extra paring computation and cost of re-randomization have to be done as compared to the CPA-decryption. Altogether, three extra pairing computations and cost of re-randomization are involved in the CCA-decryption.

The linear property of the pair encodings guarantees the re-randomization of the keys in the decryption process. The re-randomization simply ensures that decrypt queries can be answered using freshly generated keys each time during the simulation of security games.

3.1. CCA-Secure Construction from Pair Encodings. We explore a direct CCA-secure construction of predicate encryptions from the pair encodings. This construction efficiently extends the original CPA-construction of [1] to CCA-secure construction. Using this construction, we achieve CCA security of all the predicate encryptions found in [1, 5, 48] directly from the pair encodings of [1, 5, 48] with almost the same cost of CPA construction of [1]. In fact, the difference between the construction of ours and [1] is that, we use an extra component in ciphertext, three extra paring computations and one re-randomization in the decryption process.

Terminology: For fixed \( \theta_1, \theta_2, h \in \mathbb{Z}_N \) and \( h \in \mathbb{Z}_N^\alpha \), we define \( h_M := (\theta_1, \theta_2, h) \), \( \theta := (\theta_1, \theta_2, h) \) and \( c_0(\theta, \theta) := z(\theta_1 h + \theta_2) \), where \( z \) is an independent variable. Note that \( \theta_1, \theta_2, h \) and \( h \) will be understood from the context. For \( (c_y, \omega_2) \leftarrow \) Enc2\((y, N)\), we define \( c_M := (c_0, c_y) \), so \( |c_M| = \omega_1 + 1 \) if \( |c_y| = \omega_1 \). We can write \( c_M(s, h_M) := (c_0(s_0, \theta), c_y(s, h)) \) for \( s := (s_0, \ldots, s_{\omega_2}) \in \mathbb{Z}_N^{\omega_2+1}\).

Let \( P := (\text{Param}, \text{Enc1}, \text{Enc2}, \text{Pair}) \) be a primitive pair encoding scheme with the following condition (already defined in Conditions 2.7): For some \( i \in [\omega_1] \), \( c_{y,r}(s, h) = s_0 \). W.l.o.g, we assume that \( c_{y,1}(s, h) = s_0 \).

- Setup\((1^\kappa, j)\): It executes \( J := (N := p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}_{\text{bg}}(1^\kappa) \) and chooses \( g \leftarrow \mathbb{G}_T ; Z_3 \leftarrow \mathbb{G}_p \). Then runs \( n \leftarrow \mathcal{G}(j) \) and picks \( h \leftarrow \mathbb{Z}_N^\alpha \). Again picks \( \alpha, \theta_1, \theta_2 \leftarrow \mathbb{Z}_N \) and sets \( h_M := (\theta_1, \theta_2, h) \in \mathbb{Z}_N^{\omega_2+1} \). Let \( H : \{0,1\}^* \rightarrow \mathbb{Z}_N \) be a hash function. The public parameters and master secret key are given by

\[
\mathcal{PP} := (J, g, g^h, g^\alpha_T := e(g, g)\alpha, Z_3, H), \quad \mathcal{MSK} := \alpha.
\]

- KeyGen\((\mathcal{PP}, \mathcal{MSK}, x)\): It runs \( (k_x, m_2) \leftarrow \text{Enc1}(x, N) \). Let \( |k_x| = m_1 \). Picks \( r \leftarrow \mathbb{Z}_N^{m_2} \) and \( R_3 \leftarrow \mathbb{G}_{p_3}^{m_1} \). It outputs the secret key

\[
SK_x := (x, K_x := g^{k_x(s, h)} \cdot R_3).
\]

- Enc\((\mathcal{PP}, m, y)\): It runs \( (c_y, \omega_2) \leftarrow \text{Enc2}(y, N) \) and picks \( s := (s_0, \ldots, s_{\omega_2}) \leftarrow \mathbb{Z}_N^{\omega_2+1} \). Then computes \( C_{\text{cpa}} := (y, C_y := g^{c_y(s, h)}; C_{\text{INT}} := m \cdot g^\alpha_T) \), and \( h := H(C_{\text{cpa}}) \). It sets \( c_M(s, h_M) := (c_0(s_0, \theta), c_y(s, h)) \in \mathbb{Z}_N^{\omega_2+1} \), where \( |c_y| = \omega_1 \), \( \theta := (\theta_1, \theta_2, h) \), and \( c_0(s_0, \theta) := s_0(\theta_1 h + \theta_2) \). Returns \( \text{CT} := (y, C_y := g^{c_y(s, h)}; C_{\text{INT}}) \).

- Decrypt\((\mathcal{PP}, \text{CT}, SK_x)\): Let \( SK_x := (x, K_x) \). It runs \( K_x \leftarrow \text{Re-Rand}(\tilde{K}_x) \). It parses CT as \( (y, C_y := (C_0, C_y), C_{\text{INT}}) \) with \( C_0 = g^{c_0(s_0, \theta)} \) and \( C_y = g^{c_y(s, h)} \cdot \tilde{R}_3 \).
$g^{c_0(s,h)}$. Then sets $C_{\text{cpa}} := (y, C_y, C_{\text{INT}})$ and computes $h := H(C_{\text{cpa}})$. It chooses $R \leftarrow U \mathbb{G}_{p_3}$. If $x \neq y$ or $e(g \cdot R, C_0) \neq e(g^{\theta_1 h + \theta_2}, C_1)$, returns $\perp$. Otherwise, it sets $SK^x_M := (K_0, \Psi \cdot K^F_x) \in \mathbb{G}_{p_3}^{2w_1 + 1}$, where $K_0 := g^{-\tau} \cdot R_0$, $\Psi := g^\psi$ with $\psi := (\tau(\theta_1 h + \theta_2), 0, \ldots, 0) \in \mathbb{Z}_N^{w_1}$, $\tau \leftarrow U \mathbb{Z}_N$, $R_0 \leftarrow U \mathbb{G}_{p_3}$ and $E \leftarrow \text{Pair}(x, y)$. It returns $C_{\text{INT}}/e(SK^x_M, C^M_y)$.

Correctness: Let $\Delta := e(SK^x_M, C^M_y)$. For $x \sim_N y$ ($\Rightarrow x \sim_{p_1} y$ by domain transferability), we have

$$\Delta = g_T^{<(-\tau, \psi+k_x(\alpha, r, h)E), c^M_y(s, h_M)>} \quad \text{(by orthogonality of CBG)}$$

$$= g_T^{<-\tau, 0,0,0> + (0, \psi) + (0, k_x(\alpha, r, h)E), c^M_y(s, h_M)>} \quad \text{(by linearity)}$$

$$= g_T^{-(\tau c_0(s, \theta) + \tau(\theta_1 h + \theta_2) c_{y,1}(s, h) + c_x(s, h_M))}$$

$$= g_T^{-(\tau s_0(\theta_1 h + \theta_2) + \tau s_0(\theta_1 h + \theta_2) + k_x(\alpha, r, h)E c_{y,1}^T(s, h_M))} \quad \text{(since $c_{y,1}(s, h) = s_0$)}$$

$$= g_T^{\alpha s} \quad \text{(by correctness of P)}$$

Remark 4. Note that the CCA secure ciphertext can also be represented as $CT = (C_{\text{cpa}}, C_0)$, where $C_{\text{cpa}}$ is the CPA-ciphertext of [1], then followed by the computation of $C_0$.

Remark 5. The key $SK^x_M$ defined in Decrypt, we call the alternative key (in short alt-key). Using this alternative key if we run AltDecrypt (defined later), we have the same message as in Decrypt using the original key $SK^x_x$.

3.2. Security of the Proposed Construction. The core technique used to prove the adaptive CCA-security of the proposed construction in Section 3.1 is the dual system methodology of Waters [45]. Attrapadung [1] abstracted this methodology to prove adaptive security of the CPA-construction based on pair encoding scheme. Our proof technique follows the dual system CPA-proof style of [1], but it extends from CPA-proof style to CCA-proof style. In addition to answering the various queries in CPA-proof, a simulator has to answer different decrypt queries made by adversary in CPA-proof. In this style, the original adaptive CCA-game is changed to the final game through some intermediate games. These changes are made under three subgroup decision problems and (CMH and SMH) or PMH-security of the underlying pair encoding scheme.

To smooth hybrid arguments over the consecutive games, we use the natural restrictions defined in Conditions 2.7. We note that the condition (2) is only used (in Lemma 3.13) for reaching to the final game from the previous game. We use the abbreviation ‘sf-type’ for semi-functional type. For all the games, we define the semi-functional keys, ciphertexts and alt-keys of various type as follow:

- SFSetup$(1^n, j)$: It runs $(PP, MSK) \leftarrow \text{Setup}(1^n, j)$ and in addition it returns semi-functional parameters, $g_2 \leftarrow U \mathbb{G}_{p_2}$, $\hat{\theta}_1, \hat{\theta}_2 \leftarrow U \mathbb{Z}_N$ and $\hat{h} \leftarrow U \mathbb{Z}_N$. We set $h_M := (\hat{\theta}_1, \hat{\theta}_2, \hat{h})$. 

---
- SFKeyGen(PP, MSKx, g2, type, h, h): It runs (kx, m2) ← Enc1(x, N) with |kx| = m1. It chooses \( r, \tilde{r} \leftarrow Z_N^{m2} \) and \( R_3 \leftarrow Z_{p3}^{m2} \). It outputs the semi-functional key \( SK_x := (x, K_x) \), where \( K_x \) is given by:

\[
K_x := \begin{cases} 
    g_{k^x(r, r, h)} \cdot g_2^{k^x(0, r, h)} \cdot R_3 & \text{if type } = 1 \\
    g_{k^x(r, r, h)} \cdot g_2^{k^x(0, \tilde{r}, h)} \cdot R_3 & \text{if type } = 2 \\
    g_{k^x(r, r, h)} \cdot g_2^{k^x(\tilde{r}, 0, 0)} \cdot R_3 & \text{if type } = 3.
\end{cases}
\]

- SFEnc(PP, m, y, g2, type, h,M): It runs (cy, \( \omega_2 \)) ← Enc2(y, N) and picks \( s := (s_0, \ldots, s_\omega_2) \), \( \hat{s} := (\hat{s}_0, \ldots, \hat{s}_\omega_2) \). Computes \( e_M^y(s, h_M) := (c_0(s_0, \theta), e_y(s, h)) \in G_\omega +1 \) and \( e_M^y(\hat{s}, h_M) := (c_0(\hat{s}_0, \tilde{\theta}), e_y(\hat{s}, \tilde{\theta})) \in G_\omega +1 \), where \( |c_0| = \omega_1, \theta := (\theta_1, \theta_2, h), \tilde{\theta} := (\tilde{\theta}_1, \tilde{\theta}_2, h) \). It picks \( g_t \leftarrow G_T \) and returns the following semi-functional ciphertext CT:

\[
CT := \begin{cases} 
    (y, C_y^M := g_{cy}^{e_M^y(s, h_M)} \cdot g_2^{g_m^y(\hat{s}, h_M)}, C_{INT} := m \cdot g_{s_0}^\omega) & \text{if type } = 1 \\
    (y, C_y^M := g_{cy}^{e_M^y(s, h_M)} \cdot g_2^{g_m^y(\hat{s}, h_M)}, C_{INT} := m \cdot g_t^\omega) & \text{if type } = 2.
\end{cases}
\]

- AltKeyGen(PP, MSK, CT, x): Parses CT as (C_cpa, C0), computes \( h = H(C_{cpa}) \) and picks \( \tau \leftarrow Z_N, R_0 \leftarrow Z_{p3} \). It first generates the normal key, \( SK_x := (x, K_x := g_{k^x(r, r, h)} \cdot R_3) \). Then, it creates the alt-key \( SK_x^M := (K_0, \Psi, K_x^M) \in G_\omega +1 \), where \( K_0 := g_{-1} \cdot R_0, \Psi := g^2 \circ (\tau(h + \tilde{h}_2), 0, \ldots, 0) \in Z_N^{\omega+1} \) and \( E \leftarrow \text{Pair}(x, y) \).

- SFAltKeyGen(PP, MSK, CT, x, g2, type): First, it creates a normal alt-key \( SK_x^M := \text{AltKeyGen}(PP, MSK, CT, x) \). Then it picks \( b, i \leftarrow Z_N \) and returns the semi-functional alt-key \( SK_x^M \cdot g_2^\omega \), where \( \dot{\psi} \in Z_N^{\omega+1} \) is given by:

\[
\dot{\psi} := \begin{cases} 
    (b, i, 0, \ldots, 0) & \text{if type } = 1 \\
    (0, i, 0, \ldots, 0) & \text{if type } = 2.
\end{cases}
\]

- AltDecrypt(PP, CT, SK,M): This is same as Decrypt algorithm, but here we do not need to compute the alt-key as it is supplied. For sake of completeness:

It picks \( R \leftarrow Z_{p3} \). If \( x \neq y \) or \( e(g \cdot R, C_0) \neq e(g^{\theta_1}, h + \theta_2, C_1) \), it returns \( \bot \) else \( C_{INT} / e(SK_x^M, C_y^M) \).

For having a desired type of semi-functional keys (resp. alt-keys and ciphertexts), set the value of type in the arguments of SFKeyGen (resp. SFAltKeyGen and SFEnc). For example, if we set \( \text{type } = 1 \) in the arguments of SFKeyGen, we will have sf-type 1 key. There is no g2 component in the normal form of key, alt-key and challenge ciphertext, but their semi-functional variants contain g2 component. To compute semi-functional objects, a semi-functional component is composed with the normal objects. The semi-functional component is a kind of mimicry of the \( G_1 \)-structure of the normal objects into the \( G_2 \) subgroup. Some other additional changes are made depending on the type of the semi-functional object. For example, in sf-type 1 (resp. 2 and 3) key, the exponent of g2 is \( k_x(0, r, \tilde{h}) \) (resp. \( k_x(\tilde{h}, 0, 0) \)). Similar illustration holds for semi-functional alt-keys and challenge ciphertext. The difference between sf-type 1 and sf-type 2 challenge ciphertexts is the distribution
of $C_{\text{INT}}$. In former case, $C_{\text{INT}}$ is the same as that of normal ciphertext and in the later case, $C_{\text{INT}}$ is $m \cdot g_t$, where $g_t$ is a random element of $G_T$.

**Theorem 3.1.** Let $P$ be a pair encoding scheme for a predicate $\sim$ which satisfies Conditions 2.7 and $\sim$ is domain-transferable. Suppose $P$ has both the security, SMH and CMH, the assumptions, DSG1, DSG2 and DSG3 hold in $J$ and $H$ is a collision resistant hash function, then the proposed predicate encryption scheme $PE$ in Section 3.1 for the predicate $\sim$ is AP-IND-CCA secure (Definition 2.6).

**Proof.** Suppose there are at most $q$ (resp. $\nu$) key (resp. decrypt) queries made by an adversary $\mathcal{A}$. Then the security proof consists of hybrid argument over a sequence of $3q_1 + 2\nu + 7$ games, where among the $q$ key queries, $q_1$ is the number of phase-1 key queries. Let $\text{Game}_{\text{Real}}$ be the original AP-IND-CCA security game of predicate encryption scheme. By applying hybrid arguments on $\text{Game}_{\text{Real}}$ through the sequence of intermediate games $\text{Game}_{\text{Res}}$, $\text{Game}_0$, $\{\text{Game}_{1-k-1}, \text{Game}_{1-k-2}, \text{Game}_{1-k-3}\}_{k \in [q_1]}$, $\text{Game}_{1-(q_1+1)-1}$, $\text{Game}_{1-(q_1+1)-2}$, $\text{Game}_{1-(q_1+1)-3}$, $\{\text{Game}_{2-k-1}, \text{Game}_{2-k-2}\}_{k \in [\nu]}$, we reach to $\text{Game}_{\text{Final}}$. All the games are described in details in Figures 3 and 4, where the expression in the ‘box’ indicates the modification from the previous game. From $\text{Game}_0$ onwards, $\text{SFSetup}(1^\kappa, j)$ is run in setup phase to output additional components $g_2$ and $H_M$ required for generating semi-functional components. For simplicity, $\mathcal{PP}$ and $\mathcal{MSK}$ are omitted from the respective algorithms appeared in the figure.

In $\text{Game}_{\text{Res}}$, the natural restriction $x \not\sim_N y^*$ is replaced by $x \not\sim_{p_2} y^*$ for each key query $x$ made by $\mathcal{A}$. This game change is taken care of by Lemma 3.2 under DSG2 assumption. In the lemma, domain-transferability is used explicitly. In $\text{Game}_0$, the challenge ciphertext is changed from normal to sf-type 1. This change is made using Lemma 3.3 under DSG1 assumption. In $\text{Game}_{1-k-\iota}$ (for $1 \leq \iota \leq 3$), the challenge ciphertext is sf-type 1, the first $(k-1)$ keys are of sf-type 3, $k^{th}$ key is of sf-type $\iota$ and the rest keys are normal, and all the decrypt queries are answered using normal alt-keys. There are $3q_1$ game changes from $\text{Game}_0 (= \text{Game}_{1-q_1-3})$, through $\text{Game}_{1-1-1}$, $\text{Game}_{1-1-2}$, $\text{Game}_{1-1-3}$, $\text{Game}_{1-2-1}$, ..., $\text{Game}_{1-q_1-3}$. In each subsequent game change, the $k^{th}$ key is changed either from normal to sf-type 1 or sf-type 1 to sf-type 2 or sf-type 2 to sf-type 3. Note that to answer each phase-1 key query $x_i$ of the form sf-type 2 or sf-type 3, a fresh $\alpha_i$ is chosen each time. The game change, where the $k^{th}$ key gets transformed from normal to sf-type 1 (resp. sf-type 2 to sf-type 3) is argued by Lemma 3.4 (resp. 3.6) under DSG2 assumption. The hybrid argument for changing the $k^{th}$ key from sf-type 1 to sf-type 2 is assured by Lemma 3.5 under CMH security of pair encodings. Thus in $\text{Game}_{1-q_1-3}$, first $q_1$ keys become sf-type 3 and rest keys are normal. This denotes completion of translation of the phase-1 key queries into semi-functional domain.

In $\text{Game}_{1-(q_1+1)-\iota}$ (for $1 \leq \iota \leq 3$), the challenge ciphertext is sf-type 1, all the phase-1 key queries are of sf-type 3 and all the phase-2 key queries are of sf-type $\iota$, and all the decrypt queries are answered using normal alt-keys. Note that to answer all the phase-2 key queries $x_i$ of the form sf-type 2 or sf-type 3, only a single $\alpha_i$ is chosen. From $\text{Game}_{1-(q_1+1)-3}$ to $\text{Game}_{1-(q_1+1)-1}$, all the phase-2 keys are changed at once from normal to sf-type 1. This game change is taken care of by Lemma 3.7 under DSG2 assumption. From $\text{Game}_{1-(q_1+1)-1}$ (resp. $\text{Game}_{1-(q_1+1)-2}$) to $\text{Game}_{1-(q_1+1)-2}$ (resp. $\text{Game}_{1-(q_1+1)-3}$), all the phase-2 keys are changed at once from sf-type 1 (resp. sf-type 2) to sf-type 2 (resp. sf-type 3). These last two game changes are argued by Lemma 3.8 under SMH security of pair encodings and Lemma 3.9 under DSG2 assumption respectively.
| Game      | Challenge Ciphertext | Key               |
|-----------|-----------------------|-------------------|
| Real      | $\text{Enc}(m, y^*)$  | $\text{KeyGen}(x_i)$ |
| $\text{Gen}$ | $\text{Enc}(m, y^*)$  | $\text{KeyGen}(x_i)$ |
| $0$       | $\text{SFEnc}(m, y^*, g_2, 1, \hat{h}_U)$ | $\text{KeyGen}(x_i)$ |
| $1-k-1$   | $\text{SFEnc}(m, y^*, g_2, 1, \hat{h}_U)$ | $\hat{\alpha}_i \leftarrow Z_N \forall i \in [k-1]$; $\text{SFKeyGen}(x_i, g_2, 3, \hat{\alpha}_i, 0)$ if $i < k$; $\text{SFKeyGen}(x_i, g_2, 1, 0, \hat{h}_U)$ if $i = k$; $\text{KeyGen}(x_i)$ if $i > k$ |
| $1-k-2$   | $\text{SFEnc}(m, y^*, g_2, 1, \hat{h}_U)$ | $\hat{\alpha}_i \leftarrow Z_N \forall i \in [k]$; $\text{SFKeyGen}(x_i, g_2, 3, \hat{\alpha}_i, 0)$ if $i < k$; $\text{SFKeyGen}(x_i, g_2, 2, \hat{\alpha}_i, \hat{h}_U)$ if $i = k$; $\text{KeyGen}(x_i)$ if $i > k$ |
| $1-k-3$   | $\text{SFEnc}(m, y^*, g_2, 1, \hat{h}_U)$ | $\hat{\alpha}_i \leftarrow Z_N \forall i \in [k]$; $\text{SFKeyGen}(x_i, g_2, 3, \hat{\alpha}_i, 0)$ if $i < k$; $\text{KeyGen}(x_i)$ if $i > k$ |
| $1-(q_1 + 1)-1$ | $\text{SFEnc}(m, y^*, g_2, 1, \hat{h}_U)$ | $\hat{\alpha}_i \leftarrow Z_N \forall i \in [q_1]$; $\text{SFKeyGen}(x_i, g_2, 3, \hat{\alpha}_i, 0)$ if $i < q_1$; $\text{SFKeyGen}(x_i, g_2, 1, 0, \hat{h}_U)$ if $q_1 < i \leq q$ |
| $1-(q_1 + 1)-2$ | $\text{SFEnc}(m, y^*, g_2, 1, \hat{h}_U)$ | $\hat{\alpha}_i \leftarrow Z_N \forall i \in [q_1]$; $\hat{\alpha} \leftarrow Z_N$; $\text{SFKeyGen}(x_i, g_2, 3, \hat{\alpha}_i, 0)$ if $i < q_1$; $\text{SFKeyGen}(x_i, g_2, 2, \hat{\alpha}, \hat{h}_U)$ if $q_1 < i \leq q$ |
| $1-(q_1 + 1)-3$ | $\text{SFEnc}(m, y^*, g_2, 1, \hat{h}_U)$ | $\hat{\alpha}_i \leftarrow Z_N \forall i \in [q_1]$; $\hat{\alpha} \leftarrow Z_N$; $\text{SFKeyGen}(x_i, g_2, 3, \hat{\alpha}_i, 0)$ if $i < q_1$; $\text{SFKeyGen}(x_i, g_2, 3, \hat{\alpha}, 0)$ if $q_1 < i \leq q$ |

**Figure 3.** The description of the first $(3q_1 + 6)$-hybrid games used in the security proof, where alt-keys are answered by $\text{AltKeyGen}(\text{CT}_j, x_j)$. The rest of the games are described in Figure 4.

In Game${}_{2-k-i}$ (for $1 \leq i \leq 2$), the challenge ciphertext is sf-type 1, all the key queries are of sf-type 3 and the first $(k - 1)$ decrypt queries are answered using sf-type 3 alt-keys. $k^{th}$ decrypt query is answered using the alt-key of sf-type $i$ and the rest decrypt queries are answered using normal alt-keys. There are $2\nu$ game changes from Game${}_{1-(q_1+1)-3}$ (= Game${}_{2-0-2}$), through Game${}_{2-1-1}$, Game${}_{2-1-2}$, Game${}_{2-1-1}$, ..., Game${}_{2-\nu-2}$. In each subsequent game change, the $k^{th}$ alt-key is changed either from normal to sf-type 1 or sf-type 1 to sf-type 2. The game change, where the $k^{th}$ alt-key gets transformed from normal to sf-type 1 is argued by Lemma 3.10 under collision resistant property of hash and DSG2 assumption. The hybrid argument for changing the $k^{th}$ alt-key from sf-type 1 to sf-type 2 is assured by Lemma 3.12.
under DSG2 assumption. The description of Game\(_{Final}\) is the same as Game\(_{2\nu-2}\), except the challenge ciphertext is sf-type 2. The game change from Game\(_{2\nu-2}\) to Game\(_{Final}\) is done by Lemma 3.13 under DSG3 assumption.

Since the challenge ciphertext is of sf-type 2 in Game\(_{Final}\), the challenge message \(m_b\) gets masked with an independently and uniformly chosen element from \(G_T\). This implies that the component \(C_{\text{INT}}\) does not leak any information of \(b\). Therefore, the adversary \(\mathcal{A}\) has no advantage in Game\(_{Final}\).

The complete security redunction is given by:

\[
\text{Adv}_{\mathcal{A}^\text{IND-CCA}}(\kappa) \leq \text{Adv}_{\mathcal{B}_1}^{\text{DSG2}}(\kappa) + (2q_1 + 2\nu + 3)\text{Adv}_{\mathcal{B}_2}^{\text{DSG2}}(\kappa) + q_1\text{Adv}_{\mathcal{B}_3}^{\text{CMH}}(\kappa) + \text{Adv}_{\mathcal{B}_4}^{\text{CRH}}(\kappa) + \text{Adv}_{\mathcal{B}_5}^{\text{DSG3}}(\kappa)
\]

where \(\text{Adv}_{\mathcal{B}_i}^{\text{CRH}}(\kappa)\) is the advantage of \(\mathcal{B}_i\) in breaking the collision resistant property of \(H\) and, \(\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4, \mathcal{B}_5, \mathcal{B}_6\) are PPT algorithms whose running times are same as that of \(\mathcal{A}\). This completes the proof.

\[\square\]

**Lemma 3.2.** Game\(_{\text{Real}}\) and Game\(_{\text{Res}}\) are indistinguishable under the DSG2 assumption. That is, for every adversary \(\mathcal{A}\), there exists a PPT algorithm \(\mathcal{B}\) such that

\[
|\text{Adv}_{\mathcal{A}^\text{IND-CCA}}^\text{Real}(\kappa) - \text{Adv}_{\mathcal{A}^\text{IND-CCA}}^\text{Res}(\kappa)| \leq \text{Adv}_{\mathcal{B}}^{\text{DSG2}}(\kappa).
\]

**Proof.** Suppose an adversary can distinguish the games with a non-negligible probability. Then we will establish a PPT simulator \(\mathcal{B}\) for breaking the DSG2 assumption with the same probability. An instance of DSG2, \((J, g, Z_1, Z_2, W_1, W_2, Z_3, T_\beta)\) with \(\beta \leftarrow \{0, 1\}\) is given to \(\mathcal{B}\). The only difference between the games, Game\(_{\text{Real}}\) and Game\(_{\text{Res}}\) is that if \(x\) is a queried key-index and \(y^*\) is a challenge associated data-index, then it holds: \(x \sim_{p_2} y^*\) but, \(x \not\sim_{N} y^*\). We show that the above scenario will not happen. In fact, from the soundness of domain-transferability of \(\sim\), we can find a factor \(F\) such that \(p_2|F|N\). There are three possibilities of \(F\): (1) \(F = p_2\), (2) \(F = p_1p_2\) and (3) \(F = p_2p_3\). We remark the aforesaid cases are recognized using the parameters of the given instance of DSG2. Suppose \(F = p_2\). Let \(B := N/F = p_1p_3\) and then by checking \((T_\beta)^B \equiv 1\), \(\mathcal{B}\) can break the DSG2 assumption. Now suppose \(F = p_1p_2\) or \(F = p_2p_3\). Let \(B := N/F\). If \(B = p_3\), it computes \(Y_2 := (W_2W_3)^B = W_2^p\) else \(Y_2 := (Z_1Z_2)^B = Z_2^p\). In both case, we have \(Y_2 \in G_{p_2}\), then by checking \(e(T_\beta, Y_2)^* = 1\), \(\mathcal{B}\) can break the DSG2 assumption. \(\square\)
Lemma 3.3. Game_{Res} and Game_{0} are indistinguishable under the DSG1 assumption. That is, for every adversary A', there exists a PPT algorithm B such that

$$|\text{Adv}_{A',PE}^{\text{Res}}(\kappa) - \text{Adv}_{A',PE}^{0}(\kappa)| \leq \text{Adv}_{B}^{\text{DSG1}}(\kappa).$$

Proof. We establish a PPT simulator B who receives an instance of DSG1, (J, g, Z_{3}, T_{\beta}) with \beta \leftarrow \{0, 1\} and depending on the distribution of \beta, it simulates either Game_{Res} or Game_{0}.

Setup: B chooses \alpha, \theta_{1}, \theta_{2}, Z \leftarrow \mathbb{Z}_{N}, \ h \leftarrow \mathbb{Z}_{N} and sets \( h_{M} := (\theta_{1}, \theta_{2}, h) \). Let H : \{0, 1\}^{*} \rightarrow \mathbb{Z}_{N} be a hash function. Then, it provides \( PP := (J, g, g^{h_{M}}, g^{h} := e(g, g)^{\alpha}, Z_{3}, H) \) to A' and keeps \( MSK := (\alpha) \) to itself. It implicitly sets \( h_{M} := h_{M} \mod p_{2} \). By Chinese Remainder Theorem (CRT), \( h_{M} \) is independent from \( h_{M} \mod p_{1} \) and so \( h_{M} \) is perfectly distributed.

Query Phase-1: It consists of the following queries in adaptive manner:

- **KeyGen(x)**: It is a query for normal key. B can handle the key query of A', since the MSK is known to him.
- **Decrypt(CT, x)**: It is answered by running AltDecrypt algorithm on PP, CT and alt-key SK_{M} of normal form. B can compute the alt-key of normal form as the MSK is known to him.

Challenge Phase: A' provides two equal length messages \( m_{0}, m_{1} \) and a challenge index \( y^{*} \) to B. Then, B picks \( b \leftarrow \{0, 1\} \). Runs \((c_{y^{*}}, \omega_{2}) \leftarrow \text{Enc2}(y^{*}, N)\) with \( |c_{y^{*}}| = \omega_{1} \) and picks \( s' := (s_{0}', \ldots, s_{\omega_{2}}') \leftarrow \mathbb{Z}_{N}^{\omega_{2}+1} \). It first computes \( C_{\text{cpa}} := (y^{*}, C_{y^{*}} := (T_{\beta})_{\alpha}, C_{\text{INT}} := m_{b}e(g^{\alpha}, T_{\beta})^{s'} \) and then computes \( h^{*} := H(C_{\text{cpa}}) \). Finally, returns the challenge ciphertext \( CT^{*} := (y^{*}, C_{y^{*}}^{M} := (C_{0}^{M}, C_{y^{*}}^{M}), C_{\text{INT}}) \), where \( C_{0}^{M} := (T_{\beta})_{\alpha}^{s}(\theta_{1}h^{*}+\theta_{2}) \).

Query Phase-2: Similar to Query Phase-1.

Guess: A' sends a guess \( b' \) to B. If \( b' = b \) then B returns 1 else 0.

Analysis: We will show that all the objects are perfectly distributed as required. B implicitly sets \( g^{1} := T_{\beta} |_{s_{1}} \) and for \( \beta = 1, g_{2}^{1} := T_{\beta} |_{s_{2}} \). Then by linearity of P, we have \( g^{h_{M}}(s', h_{M}) = g^{h_{M}}(s_{1}s', h_{M}) \) and \( g_{2}^{h_{M}}(s', h_{M}) = g_{2}^{h_{M}}(s_{2}s', h_{M}) \). B implicitly sets \( s := t_{1}s' \mod p_{1} \) and for \( \beta = 1, s := t_{2}s' \mod p_{2} \). By CRT, \( s' \mod p_{1} \) is independent from \( s' \mod p_{2} \) and therefore \( s \) and \( s \) are perfectly distributed as required. Altogether, we have that the joint distribution of all the objects simulated by B is identical to that of Game_{Res} if \( \beta = 0 \) else Game_{0}.

Lemma 3.4. Game_{1-(k-1)} and Game_{1-k-1} are indistinguishable under DSG2 assumption. That is, for every adversary A', there exists a PPT algorithm B such that

$$|\text{Adv}_{A',PE}^{1-(k-1)}(\kappa) - \text{Adv}_{A',PE}^{1-k-1}(\kappa)| \leq \text{Adv}_{B}^{\text{DSG2}}(\kappa)$$

for \( 1 \leq k \leq q_{1} \), where \( q_{1} \) is the number of phase-1 key queries.

Proof. We establish a PPT simulator B who receives an instance of DSG2, (J, g, Z_{1}Z_{2}, W_{2}W_{3}, Z_{3}, T_{\beta}) with \( \beta \leftarrow \{0, 1\} \) and depending on the distribution of \( \beta \), it simulates either Game_{1-(k-1)} or Game_{1-k-1}. 


Setup: $\mathcal{B}$ chooses $\alpha, \theta_1, \theta_2 \leftarrow \mathbb{U}_{\mathbb{Z}_N}$, $h \leftarrow \mathbb{U}_{\mathbb{Z}_N^*}$ and sets $h_M := (\theta_1, \theta_2, h)$. Let $H : \{0,1\}^* \rightarrow \mathbb{Z}_N$ be a hash function. Then, it provides $\mathcal{P} \mathcal{P} := (J, g, g_M^{'}, g_2^{'}, e(g, g)^{\alpha}, Z_3, H)$ to $\mathcal{A}$ and keeps $\mathcal{M} \mathcal{S} \mathcal{K} := (\alpha)$ to itself. It implicitly sets $h_M := h_M \mod p_2$. By CRT, $h_M$ is independent from $h_M \mod p_1$ and so $h_M$ is perfectly distributed.

Query Phase-1: It consists of the following queries in adaptive manner:

- **KeyGen**($x$): Let $x_j$ be the $j^{th}$ query key-index. $\mathcal{B}$ answers the key $SK_{x_j}$ as follows:
  - If $j > k$, then $\mathcal{B}$ runs the KeyGen algorithm and gives the normal key to $\mathcal{A}$.
  - If $j < k$, then it is of sf-type 3 key. $\mathcal{B}$ runs $(k_{x_j}, m_2) \leftarrow$ Enc1($x_j, N$) with $|k_{x_j}| = m_1$. Picks $\alpha_j^{'}, r_j \leftarrow \mathbb{U}_{\mathbb{Z}_{m_2}}$, and $R_3 \leftarrow \mathbb{G}_{p_3}^m$. It computes the sf-type 3 key as defined below:

\[
SK_{x_j} := g^{k_{x_j} \alpha_j \cdot r_j \cdot h} \cdot (W_2 W_3 \cdot k_{x_j}^{(0, 0)} \cdot R_3).
\]

It implicitly sets $\hat{\alpha}_j := w_2 \alpha_j^{'},$ where $W_2 W_3 = g_2^w g_3^u$. So, $SK_{x_j}$ is properly distributed sf-type 3 key.

- If $j = k$ then it is either normal or sf-type 1 key. $\mathcal{B}$ runs $(k_{x_k}, m_2) \leftarrow$ Enc1($x_k, N$) with $|k_{x_k}| = m_1$. Picks $r_k^{'}, \hat{r}_k \leftarrow \mathbb{U}_{\mathbb{Z}_{m_2}}$ and $R_3 \leftarrow \mathbb{G}_{p_3}^m$. It generates the following $SK_{x_k}$ using $T_3$ of the instance of DSG2:

\[
SK_{x_k} := g^{k_{x_k} \alpha \cdot r_k \cdot h} \cdot T_3^k \cdot R_3.
\]

$\mathcal{B}$ implicitly sets $g^{t_1} := T_{3}|_{cpa}$ and for $\beta = 1, g^{t_2} := T_{\beta}|_{cpa}$. Then by linearity of $P$, we have $g^{k_{x_k} \alpha \cdot r_k \cdot h} \cdot g^{t_1 k_{x_k} (0, r_k, h)} = g^{k_{x_k} \alpha \cdot r_k^{t_1} \cdot h}$ and $g^{t_2 k_{x_k} (0, r_k, h)} = g^{k_{x_k} \alpha \cdot r_k^{t_2} \cdot h}$. $\mathcal{B}$ implicitly sets $r_k := r_k^{t_1} + t_1 \hat{r}_k$ and $r_k : = t_2 \hat{r}_k$. Since $r_k^{t_1}$ and $\hat{r}_k$ are chosen uniformly and independently from $\mathbb{Z}_{m_2}$, then so are $r_k$ and $\hat{r}_k$. Therefore, $SK_{x_k}$ is perfectly distributed normal (resp. sf-type 1) key if $\beta = 0$ (resp. $\beta = 1$).

- **Decrypt**($CT, x$): It is answered by running AltDecrypt algorithm on $\mathcal{P}P, CT$ and alt-key $SK_{x_k}^M$ of normal form. $\mathcal{B}$ can compute the alt-key of normal form as the $\mathcal{M}SK$ is known to him.

Challenge Phase: $\mathcal{A}$ provides two equal length messages $m_0, m_1$ and a challenge index $y^\ast$ to $\mathcal{B}$. Then, $\mathcal{B}$ picks $b \leftarrow \mathbb{U}_{\{0,1\}}$. Runs $(c_{y^\ast}, \omega_2) \leftarrow$ Enc2($y^\ast, N$) with $|c_{y^\ast}| = \omega_1$ and picks $s' := (s'_0, \ldots, s'_{\omega_2}) \leftarrow \mathbb{U}_{\mathbb{Z}_{\omega_2}^{\omega_2+1}}$, and $s':= (s'_0, \ldots, s'_{\omega_2}) \leftarrow \mathbb{Z}_{\omega_2}^{\omega_2+1}$. It first computes $C_{cpa}^y := (y^*, C_{y^*} = (Z_1 Z_2)_{c_{y^*}(s', h)}, C_{INT} := m_b \cdot e(g^y, Z_1 Z_2)^{s_0})$ and then computes $h^\ast := H(C_{cpa})$. Finally, returns the challenge ciphertext $CT^\ast := (y^*, C_{y^*}^{M} := (C_0^0, C_{y^*})$, $C_{INT}$), where $C_0^0 := (Z_1 Z_2)^{s_0(\theta_1 + \theta_2)}$. If $Z_1 Z_2 = g^{y^\ast} g_2^{z_2}$, it implicitly sets $s := z_2 s' \mod p_1$ and $s := z_2 s' \mod p_2$. By CRT, we have $s' \mod p_1$ is independent from $s' \mod p_2$. Therefore, CT$^\ast$ is perfectly distributed sf-type 1 challenge ciphertext.

Query Phase-2: It consists of the following queries in adaptive manner:

- **KeyGen**($x$): $\mathcal{B}$ runs the KeyGen algorithm and gives the normal key to $\mathcal{A}$.

- **Decrypt**($CT, x$): Similar to **Query Phase-1**.

Guess: $\mathcal{A}$ sends a guess $b'$ to $\mathcal{B}$. If $b' = b$ then $\mathcal{B}$ returns 1 else 0.
Analysis: Altogether, we have that the joint distribution of all the objects simulated by $B$ is identical to that of Game$_{1-(k-1)-3}$ if $\beta = 0$ else Game$_{1-k-1}$.

Lemma 3.5. Game$_{1-k-1}$ and Game$_{1-k-2}$ are indistinguishable under CMH security of the primitive pair encoding scheme, $P$. That is, for every adversary $A$, there exists a PPT algorithm $B$ such that

$$|\text{Adv}^{1-k-1}_{A,PE}(\kappa) - \text{Adv}^{1-k-2}_{A,PE}(\kappa)| \leq \text{Adv}^{\text{CMH}}_{A,P}(\kappa) \text{ for } 1 \leq k \leq q_1.$$  

Proof. Suppose $A$ can distinguish Game$_{1-k-1}$ and Game$_{1-k-2}$ with non-negligible probability. Then we will construct a PPT simulator $B$ for breaking the CMH security of $P$ with the same probability.

Setup: The challenger $CH$ of $P$ gives $(g, g_2, g_3) \in G_{p_1} \times G_{p_2} \times G_{p_3}$ to $B$. $B$ chooses $\alpha, \theta_1, \theta_2 \leftarrow U\mathbb{Z}_N, h \leftarrow U\mathbb{Z}_N^*$ and sets $h_M := (\theta_1, \theta_2, h)$. Let $H : \{0,1\}^* \rightarrow \mathbb{Z}_N$ be a hash function. Then, it provides $PP := (J, g, g_1^h, g_1^\theta := e(g, g)^\alpha, Z_3 := g_3, H)$ to $A$ and keeps $MSK := (\alpha)$ and $g_2$ to itself.

Query Phase-1: It consists of the following queries in adaptive manner:

- **KeyGen($x$):** Let $x_j$ be the $j^{th}$ query key-index. $B$ answers the key $SK_{x_j}$ as follows:
  - If $j > k$, then $B$ runs the KeyGen algorithm and gives the normal key to $A$.
  - If $j < k$, then it is of sf-type 3 key. Using $PP$, $MSK$ and $g_2$, $B$ can generate the required key.
  - If $j = k$ then it is either of sf-type 1 or sf-type 2 key. $B$ runs $(k_{x_k}, m_2) \leftarrow \text{Enc}(x_k, N)$ with $|k_{x_k}| = m_1$. Picks $r_k \leftarrow U\mathbb{Z}_N^m$ and $R_3 \leftarrow U\mathbb{Z}_{p_3}^m$. It makes a query with $x_k$ to $CH$ and let $T := g_2^{k_{x_k}(\beta, r_k, h)}$ be the reply, where $\beta = 0$ or random element from $\mathbb{Z}_N$. Then $B$ returns the following key $SK_{x_k} := g^{k_{x_k}(\alpha, r_k, h)} \cdot T \cdot R_3$ to $A$. Therefore, $SK_{x_k}$ is perfectly distributed sf-type 1 key if $\beta = 0$ else sf-type 2.

- **Decrypt($CT$, $x$):** It is answered by running AltDecrypt algorithm on $PP$, $CT$ and alt-key $SK_M$ of normal form. $B$ can compute the alt-key of normal form as the $MSK$ is known to him.

Challenge Phase: $A$ provides two equal length messages $m_0, m_1$ and a challenge index $y^*$ to $B$. Then, $B$ picks $b \leftarrow U\{0,1\}$. Runs $(c_{y^*}, \omega_2) \leftarrow \text{Enc}(y^*, N)$ with $|c_{y^*}| = \omega_1$ and picks $s := (s_0, \ldots, s_{\omega_2}) \leftarrow U\mathbb{Z}_N^{\omega_2+1}$. Then, it makes a query with $y^*$ to $CH$ and let $D := g_2^{c_{y^*}(s, h)}$ be the reply. It first computes $C^*_{\text{cpa}} := (y^*, C_{y^*} := g^{c_{y^*}(s, h)}, D, C_{\text{INT}} := m_b \cdot e(g, g)^{\alpha s_0})$ and then computes $h^* := H(C^*_{\text{cpa}})$. Finally, returns the challenge ciphertext $CT^* := (y^*, C^*_{y^*} := (C^*_{0}, C_{y^*}), C_{\text{INT}})$, where $C^*_{0} := (g^{s_0 \cdot g_2^{t_0}}(\theta_1 h^* + \theta_2)).$

Query Phase-2: It consists of the following queries in adaptive manner:

- **KeyGen($x$):** $B$ runs the KeyGen algorithm and gives the normal key to $A$.
- **Decrypt($CT$, $x$):** Similar to Query Phase-1.
Guess: $A$ sends a guess $b'$ to $B$. If $b' = b$ then $B$ returns 1 else 0.

Analysis: It consists of two parts, correctness and perfectness which are described below.

- Correctness: $B$ follows the restriction of CMH security game (while interacting with $C\mathcal{H}$) as long as $A$ does so in CCA-security game with $B$. In fact, by natural restriction, for all key queries $x$ made by $A$, we have $x \not\in \{p_2, y^*\}$, in particular for $k^\text{th}$ query, $x_k \not\in \{p_2, y^*\}$. Therefore, $B$ does not violate the restriction of the CMH security game with $C\mathcal{H}$.

- Perfectness: By the assumption: $c_{y^*}(\hat{s}, \hat{h}) = s_0$, the first component of $D$ is $g_2^{s_0}$. So, $C_0^*$ can be easily computable using $g_2^{s_0}, s_0, \theta_1, \theta_2$ and $h^*$. If we set $\hat{h}_M := h_M \mod p_2$, then by CRT, we have $\hat{h}_M$ is independent from $h_M \mod p_1$. Hence, $CT^*$ can be written as $(y^*, C_M^* := g_{\hat{y}}^{c_0(\hat{s}, \hat{h}_M)} \cdot g_2^{c_1(\hat{s}, \hat{h}_M)}, C_{\text{INT}})$. Therefore, $CT^*$ is perfectly distributed sf-type 1 challenge ciphertext. Altogether, we have that the joint distribution of all the objects simulated by $B$ is identical to that of Game$_{1-k-1}$ if $\beta = 0$ else Game$_{1-k-2}$.

$\square$

**Lemma 3.6.** Game$_{1-k-2}$ and Game$_{1-k-3}$ are indistinguishable under the DSG2 assumption. That is, for every adversary $A$, there exists a PPT algorithm $B$ such that

$$|\text{Adv}^{1-k-2}_{A, \mathcal{PE}}(\kappa) - \text{Adv}^{1-k-3}_{A, \mathcal{PE}}(\kappa)| \leq \text{Adv}^{\text{DSG2}}_{B}(\kappa) \quad \text{for} \quad 1 \leq k \leq q_1.$$

**Proof.** We establish a PPT simulator $B$ who receives an instance of DSG2, $(f, g, Z_1Z_2, W_2W_3, Z_3, T_\beta)$ with $\beta \overset{\$}{\leftarrow}\{0, 1\}$ and depending on the distribution of $\beta$, it simulates either Game$_{1-k-2}$ or Game$_{1-k-3}$. Description of the simulation is same as that of Lemma 3.4 except the answering $k^\text{th}$ key query. Below, we only describes the simulation of $k^\text{th}$ query:

The $k^\text{th}$ key is either sf-type 2 or sf-type 3. $B$ runs $(k_{x_k}, m_2) \overset{\$}{\leftarrow}\text{Enc}_1(x_k, N)$ with $|k_{x_k}| = m_1$. Picks $\alpha'_k \overset{\$}{\leftarrow}\mathbb{Z}_N, r'_k, \hat{r}'_k \overset{\$}{\leftarrow}\mathbb{Z}_N^{m_2}$ and $R_3 \overset{\$}{\leftarrow}\mathbb{Z}_N^{m_1}$. It generates the following $SK_{x_k}$ using $T_\beta$ of the instance of DSG2:

$$SK_{x_k} := g^{k_{x_k}}(\alpha, r'_k, h) \cdot (W_2W_3)^{k_{x_k}}(\alpha', 0, 0) \cdot t^{k_{x_k}}_\beta(0, r'_k, h) \cdot R_3.$$  

If $W_2W_3 = g_2^{\alpha_k}g_3^{s_3}$ and $T_\beta = g_1^1g_2^2g_3^{s_3}$ (for $\beta = 1$), then $B$ implicitly sets $\hat{\alpha}_k := w_2\alpha'_k, r_k := r'_k + t_1\hat{r}'_k$ and $\hat{r}_k := t_2\hat{r}'_k$. Note that here we use the linearity and param-vanishing properties of the pair encoding scheme $P$. Since $r'_k$ and $r'_k$ are chosen uniformly and independently from $\mathbb{Z}_N^{m_2}$, then so are $r_k$ and $\hat{r}_k$. Therefore, $SK_{x_k}$ is perfectly distributed sf-type 2 (resp. sf-type 3) key if $\beta = 1$ (resp. $\beta = 0$).

$\square$

**Lemma 3.7.** Game$_{1-q_1-3}$ and Game$_{1-(q_1+1)-1}$ are indistinguishable under the DSG2 assumption. That is, for every adversary $A$, there exists a PPT algorithm $B$ such that

$$|\text{Adv}^{1-q_1-3}_{A, \mathcal{PE}}(\kappa) - \text{Adv}^{1-(q_1+1)-1}_{A, \mathcal{PE}}(\kappa)| \leq \text{Adv}^{\text{DSG2}}_{B}(\kappa).$$

**Proof.** We establish a PPT simulator $B$ who receives an instance of DSG2, $(f, g, Z_1Z_2, W_2W_3, Z_3, T_\beta)$ with $\beta \overset{\$}{\leftarrow}\{0, 1\}$ and depending on the distribution of $\beta$, it simulates either Game$_{1-q_1-3}$ or Game$_{1-(q_1+1)-1}$.
Setup: $B$ chooses $\alpha, \theta_1, \theta_2 \overset{r}{\leftarrow} \mathbb{Z}_N$, $h \overset{r}{\leftarrow} \mathbb{Z}_N^n$ and sets $h_M := (\theta_1, \theta_2, h)$. Let $H : \{0, 1\}^* \rightarrow \mathbb{Z}_N$ be a hash function. Then, it provides $\mathcal{PP} := (x, g, h^x, g_2^\beta := \epsilon(g, g)^\alpha, Z_3, H)$ to $\mathcal{A}$ and keeps $M\mathcal{SK} := (\alpha)$ to itself. It implicitly sets $h_M := h_M \mod p_2$. By CRT, $h_M$ is independent from $h_M \mod p_1$ and so $h_M$ is perfectly distributed.

Query Phase-1: It consists of the following queries in adaptive manner:

- **KeyGen($x$):** This is same as the case $j < k$ of that of Lemma 3.4. In fact, let $x_j$ be the $j^{th}$ ($j \leq q_1$) query key-index. $B$ answers the key $\mathcal{SK}_{x_j}$ as follows: It is of sf-type 3 key. $B$ runs $(k_{x_j}, m_2) \leftarrow \text{Enc1}(x_j, N)$ with $|k_{x_j}| = m_1$. Picks $\alpha_j \overset{r}{\leftarrow} \mathbb{Z}_N^2$, $r_j \overset{r}{\leftarrow} \mathbb{Z}_N^{m_2}$ and $R_3 \overset{r}{\leftarrow} G_p^{m_3}$. It computes the sf-type 3 key as defined below:

\[
\mathcal{SK}_{x_j} := g^{k_{x_j}(\alpha, r_j, h)} \cdot (W_2 W_3^{k_{x_j}(\alpha, 0, 0)}) \cdot R_3.
\]

It implicitly sets $\hat{\alpha}_j := w_2 \alpha_j'$, where $W_2 W_3 = g_2^{w_2} g_3^{w_3}$. So, $\mathcal{SK}_{x_j}$ is properly distributed sf-type 3 key.

- **Decrypt($CT, x$):** It is answered by running $\text{AltDecrypt}$ algorithm on $\mathcal{PP}$, $CT$ and alt-key $\mathcal{SK}_M$ of normal form. $B$ can compute the alt-key of normal form as the $M\mathcal{SK}$ is known to him.

Challenge Phase: This is same as that of Lemma 3.4. In fact, $\mathcal{A}$ provides two equal length messages $m_0, m_1$ and a challenge index $y^*$ to $B$. Then, $B$ picks $b \overset{r}{\leftarrow} \{0, 1\}$. Runs $(e_{y^*}, \omega_2) \leftarrow \text{Enc2}(y^*, N)$ with $|e_{y^*}| = \omega_1$ and picks $s' := (s'_0, \ldots, s'_\omega_2) \overset{r}{\leftarrow} \mathbb{Z}_N^{\omega_2 + 1}$. It first computes $C_{e_{y^*}} := (y^*, C_{y^*} := (Z_1 Z_2)^{e_{y^*}(s', h)})$, $C_{\text{INT}} := m_0 \cdot e(g^\alpha, Z_1 Z_2)^{e_{y^*}(s', h)}$ and then computes $h^* := H(C_{e_{y^*}})$. Finally, returns $CT' := (y^*, C_{e_{y^*}}^M := (C_0^*, C_{y^*}^*, C_{\text{INT}}^*))$, where $C_0^* := (Z_1 Z_2)^{e_{y^*}(0)} \cdot h^*$. If $Z_1 Z_2 = g_1^z g_2^{z'}$, it implicitly sets $s := z_1 s'$ mod $p_1$ and $\hat{s} := z_2 s'$ mod $p_2$. By CRT, we have $s'$ mod $p_1$ is independent from $s'$ mod $p_2$. Therefore, $CT'$ is perfectly distributed sf-type 1 challenge ciphertext.

Query Phase-2: It consists of the following queries in adaptive manner:

- **KeyGen($x$):** This is same as the case $j = k$ of that of Lemma 3.4. In fact, let $x_j$ be the $j^{th}$ ($j > q_1$) query key-index. $B$ answers the key $\mathcal{SK}_{x_j}$ as follows: $B$ runs $(k_{x_j}, m_2) \leftarrow \text{Enc1}(x_j, N)$ with $|k_{x_j}| = m_1$. Picks $r_j', \hat{r}_j' \overset{r}{\leftarrow} \mathbb{Z}_N^{m_2}$ and $R_3 \overset{r}{\leftarrow} G_p^{m_3}$. It computes $\mathcal{SK}_{x_j}$ as defined below:

\[
\mathcal{SK}_{x_j} := g^{k_{x_j}(\alpha, r_j', h)} \cdot (T_\beta)^{k_{x_j}(0, \hat{r}_j', h)} \cdot R_3.
\]

$B$ implicitly sets $g^{t_1} := T_\beta |_{G_p}$ and for $\beta = 1$, $g_2^{t_2} := T_\beta |_{G_p^2}$. Then by linearity of $P$, we have $g^{k_{x_j}(\alpha, r_j', h)} = g^{t_1 k_{x_j}(\alpha, r_j', h)} = g^{k_{x_j}(\alpha, r_j'+t_1 \hat{r}_j', h)}$ and $g_2^{t_2 k_{x_j}(0, \hat{r}_j', h)} = g_2^{t_2 k_{x_j}(0, t_2 \hat{r}_j', h)}$. $B$ implicitly sets $r_j := r_j' + t_1 \hat{r}_j'$ and $\hat{r}_j := t_2 \hat{r}_j'$. Since $r_j'$ and $\hat{r}_j'$ are chosen uniformly and independently from $\mathbb{Z}_N^{m_2}$, then so are $r_j$ and $\hat{r}_j$. Therefore, $\mathcal{SK}_{x_j}$ is perfectly distributed normal (resp. sf-type 1) key if $\beta = 0$ (resp. $\beta = 1$).

- **Decrypt($CT, x$):** Same as Query Phase-1.

Guess: $\mathcal{A}$ sends a guess $b'$ to $B$. If $b' = b$ then $B$ returns 1 else 0.
**Analysis:** Altogether, we have that the joint distribution of all the objects simulated by $B$ is identical to that of Game$_{1-q_1-3}$ if $\beta = 0$ else Game$_{1-(q_1+1)-1}$. □

**Lemma 3.8.** Game$_{1-(q_1+1)-1}$ and Game$_{1-(q_1+1)-2}$ are indistinguishable under the SMH security of of the primitive pair encoding scheme, $P$. That is, for every adversary $A$, there exists a PPT simulator $B$ such that

$$|\text{Adv}_{A,P}^{1-(q_1+1)-1}(\kappa) - \text{Adv}_{A,P}^{1-(q_1+1)-2}(\kappa)| \leq \text{Adv}_{SMH}^{SMH}(\kappa).$$

**Proof.** Suppose $A$ can distinguish Game$_{1-(q_1+1)-1}$ and Game$_{1-(q_1+1)-2}$ with non-negligible probability. Then we will construct a PPT simulator $B$ for breaking the SMH security of $P$ with the same probability.

**Setup:** The challenger $CH$ of $P$ gives $(g, g_2, g_3) \in G_{p_1} \times G_{p_2} \times G_{p_3}$ to $B$. $B$ chooses $\alpha, \theta_1, \theta_2 \leftarrow \mathbb{Z}_N$, $h \leftarrow \mathbb{Z}_N^\times$ and sets $h_\mathbb{M} := (\theta_1, \theta_2, h)$. Let $H : \{0, 1\}^* \rightarrow \mathbb{Z}_N$ be a hash function. Then, it provides $PP := (\mathcal{J}, g, g^h, g_\text{CP}^\theta : = e(g, g)^\alpha, Z_3 : = g_3, H)$ to $A$ and keeps $MSK := (\alpha)$ and $g_2$ to itself.

**Query Phase-1:** It consists of the following queries in adaptive manner:
- **KeyGen$(x)$:** It is of sf-type 3 key. Using $PP$, $MSK$ and $g_2$, $B$ can generate the required key.
- **Decrypt$(CT, x)$:** It is answered by running $\text{AltDecrypt}$ algorithm on $PP$, $CT$ and alt-key $SK_x^M$ of normal form. $B$ can compute the alt-key of normal form as the $MSK$ is known to him.

**Challenge Phase:** $A$ provides two equal length messages $m_0, m_1$ and a challenge index $y^*$ to $B$. Then, $B$ picks $b \leftarrow \{0, 1\}$. Runs $(c_{y^*}, \omega_2) \leftarrow \text{Enc}_2(y^*, N)$ with $|c_{y^*}| = \omega_1$ and picks $s := (s_0, \ldots, s_{\omega_2}) \leftarrow \mathbb{Z}_N^{\omega_2+1}$. Then, it makes a query with $y^*$ to $CH$ and let $D := g_2^{c_{y^*}(s, h)}$ be the reply. It first computes $C_{\text{CP}} := (y^*, C_{y^*} : = g_\text{CP}^{c_{y^*}(s, h)}, D, C_{\text{INT}} : = m_b \cdot e(g, g)^{s_{\omega_2}})$ and then computes $h^* := H(C_{\text{CP}})$. Finally, returns the challenge ciphertext $CT^* := (y^*, C_{y^*} := (C_0^*, C_{y^*}), C_{\text{INT}})$, where $C_0^* := (g^{s_0}g_2^s)_{(\theta_1, h^* + \theta_2)}$.

**Query Phase-2:** It consists of the following queries in adaptive manner:
- **KeyGen$(x)$:** All the keys are either sf-type 1 or sf-type 2. Let $x_j$ be the $j^{th}$ query key-index. $B$ runs $(k_{x_j}, m_2) \leftarrow \text{Enc}_1(x_j, N)$ with $|k_{x_j}| = m_1$. Picks $r_j \leftarrow \mathbb{Z}_N^{m_2}$ and $R_3 \leftarrow \mathbb{G}_{p_3}^{m_1}$. It makes a query with $x_j$ to $CH$ and let $T := g_2^{k_{x_j}(\beta, r, h)}$ be the reply, where $\beta = 0$ or random element from $\mathbb{Z}_N$. Then $B$ returns the following key

$$SK_{x_j} := g^{k_{x_j}(\alpha, r, h)} \cdot T \cdot R_3$$

to $A$. Therefore, $SK_{x_j}$ is perfectly distributed sf-type 1 key if $\beta = 0$ else sf-type 2.
- **Decrypt$(CT, x)$:** Similar to **Query Phase-1**.

**Guess:** $A$ sends a guess $b'$ to $B$. If $b' = b$ then $B$ returns 1 else 0.

**Analysis:** It consists of two parts, correctness and perfectness which are described below.
- **Correctness:** $B$ follows the restriction of SMH security game (while interacting with $CH$) as long as $\mathcal{A}$ does so in CCA-security game with $B$. In fact, by natural restriction, for all key queries $x$ made by $\mathcal{A}$, we have $x \not\in p_2$ $y^*$, in particular for $k^{th}$ query, $x_k \not\in p_2$ $y^*$. Therefore, $B$ does not violate the restriction of the SMH security game with $CH$.

- **Perfectness:** By the assumption: $c_{w^1}(s, h) = s_0$, the first component of $D$ is $g_2^{x_0}$. So, $C^*_0$ can be easily computable using $g_2^{x_0}$, $s_0$, $\theta_1$, $\theta_2$ and $h^*$. If we set $h_m := h_m$ mod $p_2$, then by CRT, we have $h_m$ is independent from $h_m$ mod $p_1$. Hence, $CT^*$ can be written as $(y^*, C^*_y := g_{c_1}^{H_0(a, h_m)} \cdot g_{c_2}^{H_1(a, h_m)}, C_{\text{INT}})$. Therefore, $CT^*$ is perfectly distributed sf-type 1 challenge ciphertext. Altogether, we have that the joint distribution of all the objects simulated by $B$ is identical to that of Game$_1-(q_1+1)-1$ if $β = 0$ else Game$_1-(q_1+1)-2$.

\[ \square \]

**Lemma 3.9.** Game$_1-(q_1+1)-2$ and Game$_1-(q_1+1)-3$ are indistinguishable under the DSG2 assumption. That is, for every adversary $\mathcal{A}$, there exists a PPT algorithm $B$ such that

\[ |\text{Adv}_{\mathcal{A}, PE}^{1-(q_1+1)-2}(\kappa) - \text{Adv}_{\mathcal{A}, PE}^{1-(q_1+1)-3}(\kappa)| \leq \text{Adv}_{\mathcal{A}, PE}^{\text{DSG2}}(\kappa). \]

**Proof.** We establish a PPT simulator $B$ who receives an instance of DSG2, $(\mathcal{J}, g, Z_1, Z_2, W_2, W_3, Z_3, T_2)$ with $β \overset{\$}{\leftarrow} (0, 1)$ and depending on the distribution of $β$, it simulates either Game$_1-(q_1+1)-2$ or Game$_1-(q_1+1)-3$. The simulation is almost similar to Lemma 3.7 except answering the key queries after the challenge phase. We illustrate here only the key queries after the challenge phase. Let $x_j$ be the $j^{th}$ $(j > q_1)$ query key-index. Note that for all the post key queries $x_j$, $\hat{α}$'s appearing in $G_{p_2}$ components of $SK_{x_j}$ are identical.

$B$ runs $(k_{x_j}, m_2) \leftarrow \text{Enc}(x_j, N)$ with $|k_{x_j}| = m_1$. Picks $α' \leftarrow Z_N$, $r'_j$, $t'_j \leftarrow Z_{N^2} \overset{\$}{\leftarrow} G_{p_3}$. It computes $SK_{x_j}$ as defined below:

\[
SK_{x_j} := g_{k_{x_j}(α, r'_j, h)} \cdot (W_2 W_3)^{k_{x_j}(α', 0, 0)} \cdot (T_2)^{k_{x_j}(0, r'_j, h)} \cdot R_3.
\]

Let $W_2 W_3 = g_{2}^{w_2} g_{3}^{w_3}$. $B$ implicitly sets $g^{t_1} := T_2|_{G_{p_3}}$ and for $β = 1$, $g^{t_2} := T_2|_{G_{p_2}}$. Then by linearity of $P$, we have

\[
g^{k_{x_j}(α, r'_j, h)} \cdot g^{t_1 k_{x_j}(0, \hat{r}'_j, h)} = g^{k_{x_j}(α, r'_j + t_1 \hat{r}'_j, h)}
\]

and

\[
w_2 k_{x_j}(α', 0, 0) \cdot t_2 k_{x_j}(0, \hat{r}'_j, h) = g_{2}^{k_{x_j}(w_2 α', t_2 \hat{r}'_j, h)}.
\]

$B$ implicitly sets $\hat{α} := w_2 α'$, $r_j := r'_j + t_j \hat{r}'_j$ and $\hat{r}_j := t_2 \hat{r}'_j$. Since $r'_j$ and $\hat{r}'_j$ are chosen uniformly and independently from $Z_{N^2}$, then so are $r_j$ and $\hat{r}_j$. Therefore, $SK_{x_j}$ is perfectly distributed normal (resp. sf-type 1) key if $β = 0$ (resp. $β = 1$).

\[ \square \]

**Lemma 3.10.** Game$_2-(k-1)-2$ and Game$_2-(k-1)$ are indistinguishable under the DSG2 assumption and collision resistant property of $H$. That is, for every adversary $\mathcal{A}$, there exists a PPT algorithm $B$ such that

\[ |\text{Adv}_{\mathcal{A}, PE}^{2-(k-1)-2}(\kappa) - \text{Adv}_{\mathcal{A}, PE}^{2-(k-1)}(\kappa)| \leq \text{Adv}_{\mathcal{A}, PE}^{\text{DSG2}}(\kappa) + \text{Adv}_{\mathcal{A}, PE}^{\text{CRH}}(\kappa) \text{ for } 1 \leq k \leq ν. \]
Proof. We establish a PPT simulator \( \mathcal{B} \) who receives an instance of DSG2, \((J, g, Z_1 Z_2, W_2 W_3, T_3, T_β)\) with \( β \leftarrow \{0, 1\} \) and depending on the distribution of \( β \), it simulates either Game\(_2^{-\ell(k-1)-2} \) or Game\(_2^{k-1} \).

Setup: \( \mathcal{B} \) chooses \( α, β_1, β_2 \leftarrow \mathbb{Z}_N \), \( h \leftarrow \mathbb{Z}_N^\ast \) and sets \( h_M := (β_1, β_2, h) \). Let \( H : \{0, 1\}^* \rightarrow \mathbb{Z}_N \) be a hash function. Then, it provides \( PP := (J, g, g^h_M, g_2^h := e(g, g)^α, Z_3, H) \) to \( \mathcal{A} \) and keeps \( MSK := (α) \) to itself. Implicitly sets \( h_M := h_M \mod p_2 \). By CRT, \( h_M \) is independent from \( h_M \mod p_1 \) and so \( h_M \) is perfectly distributed.

Query Phase-1: It consists of the following queries in adaptive manner:

- **KeyGen\((x)\):** Here all the keys are of sf-type 3 and simulation of the keys are same as the sf-type 3 keys of Lemma 3.4.
- **Decrypt\((CT, x)\):** Let \((CT, x)\) be the \( j^{th} \) decrypt query made \( \mathcal{A} \). \( \mathcal{B} \) first constructs the alt-key \( SK_{x_j}^M \) as shown below and then answers to \( \mathcal{A} \) by running AltDecrypt algorithm:
  - If \( j > k \), it is normal alt-key. \( \mathcal{B} \) can compute the key as the \( MSK \) is known to him.
  - If \( j < k \), it is sf-type 2 alt-key. \( \mathcal{B} \) first computes the normal alt-key \( SK_{x_j}^M \), picks \( t_j \leftarrow \mathbb{Z}_N \) and then creates the sf-type 2 alt-key as follows:
    \[
    SK_{x_j}^M := SK_{x_j}^M \cdot (W_2 W_3)^{(0, t_j, 0, \ldots, 0)}.
    \]
    If \( W_2 W_3 = g_2^{w_2} g_3^{w_3} \), then \( \mathcal{B} \) implicitly sets \( t_j := w_2 t_j' \). So, \( SK_{x_j}^M \) is properly distributed sf-type 2 alt-key.
  - If \( j = k \), it is either normal or sf-type 1 alt-key. \( \mathcal{B} \) runs \((k_{x_k}, m_2) \leftarrow \text{Enc1}(x_k, N) \) and \( \text{Pair}(x_k, y_k) \rightarrow E \in \mathbb{Z}_N^{m_2 \times w_1} \). Picks \( r \leftarrow \mathbb{Z}_N^{m_2} \) and \( R_3 \leftarrow \mathbb{G}_p^{w_1+1} \). It computes the alt-key as given below:
    \[
    SK_{x_k}^M := g^{(0, k_{x_k}(α, r, h)E)} \cdot T_{β}^{-1, 0, \ldots, 0}, T_{β}^{0, 0, \theta_1 h_k + \theta_2, \ldots, 0} \cdot R_3,
    \]
    where \( h_k := H(C_{\text{cpa}}^{(k)}) \). Let \( g^r := T_{β}^{|G_p^1|} \) and for \( β = 1 \), \( g_2^{t_2} := T_{β}^{|G_p^2|} \).
  - Then, \( \mathcal{B} \) sets \( g_r := SK_{x_k}^M \mid_{G_p^1} \), where \( v := (−τ, ψ + k_{x_k}(α, r, h)E) \) and \( ψ := (τ(θ_1 h_k + \theta_2), 0, \ldots, 0) \). If \( β = 1 \), it sets \( g_2^{t_2} := SK_{x_k}^M \mid_{G_p^2} \) with \( v := (b, t, 0, \ldots, 0) \in \mathbb{Z}_N^{w_1+1} \), where \( \mathcal{B} \) implicitly sets \( b := −t_2 \mod p_2 \) and \( t := t_2(θ_1 h_k + \theta_2) \mod p_2 \). Therefore, \( SK_{x_k}^M \) is perfectly distributed normal (resp. sf-type 1) alt-key if \( β = 0 \) (resp. \( β = 1 \)).

**Challenge Phase:** \( \mathcal{A} \) provides two equal length messages \( m_0, m_1 \) and a challenge index \( y^* \) to \( \mathcal{B} \). Then, \( \mathcal{B} \) picks \( b \leftarrow \{0, 1\} \). Runs \((c_{y'}, ω_2) \leftarrow \text{Enc2}(y^*, N) \) with \( |c_{y'}| = ω_1 \) and picks \( s' := (s'_{u_0}, \ldots, s'_{u_2}) \leftarrow U \mathbb{Z}_N^{ω_2+1} \). It first computes \( C_{\text{cpa}} := (y^*, C_{y^*} := (Z_1 Z_2)_{c_{y'}}(s', h), C_{\text{INT}} := m_b \cdot e(g^a, Z_1 Z_2)^{s'} \) and then computes \( h^* := H(C_{\text{cpa}}) \). Finally, returns the challenge ciphertext \( CT^* := (y^*, C_{y^*} := (C_0^*, C_{y^*}) \), C_{\text{INT}} \), where \( C_0^* := (Z_1 Z_2)^{s'}(θ h^* + θ_2) \). If \( Z_1 Z_2 = g_2^{s_2} g_3^{s'} \), it implicitly sets \( s := z_1 s' \mod p_1 \) and \( s := s_2 s' \mod p_2 \). By CRT, \( s' \mod p_1 \) is independent from \( s' \mod p_2 \). Therefore, \( CT^* \) is perfectly distributed sf-type 1 challenge ciphertext.

**Query Phase-2:** It consists of the following queries in adaptive manner:

\[ \]
KeyGen($x$): Again note that for all the post key queries $x$, $\hat{\alpha}$'s appearing in $G_{p_2}$ components of $SK_x$ are identical. $B$ picks $\alpha' \leftarrow U \mathbb{Z}_N$. Let $x_j$ be the $j^{th}$ ($j > q_1$) query key-index. $B$ runs $(k_{x_j}, m_2) \leftarrow \text{Enc1}(x_j, N)$ with $|k_{x_j}| = m_1$. It chooses $r_j \leftarrow U \mathbb{Z}_{N^2}^*$ and $R_3 \leftarrow U \mathbb{G}_{p_3}^m$. It computes $SK_{x_j}$ as defined below:

$$SK_{x_j} := g^{k_{x_j}(\alpha', r_j, h)} . (W_2 W_3)^{k_{x_j}(\alpha', 0, 0)} . R_3.$$ 

It implicitly sets $\hat{\alpha} := w_0 \alpha'$, where $W_2 W_3 = g^{w_0} g^{w_3}$. So, $SK_{x_j}$ is properly distributed sf-type 3 key.

Decrypt($CT, x$): Similar to Query Phase-1 except for the $k^{th}$ decrypt query $(CT_k, x_k)$, i.e., if $CT^* \neq CT_k$ and $h^* = h_k$, then $B$ aborts.

**Guess:** $A$ sends a guess $b'$ to $B$. If $b' = b$ then $B$ returns 1 else 0.

**Analysis:** By the natural restriction of the security game, $A$ is allowed to decrypt query $(CT_k, x_k)$ if $CT^* \neq CT_k$. Now, we mainly concentrate on the joint distribution of $k^{th}$ alt-key (for answering decryption query $(CT_k, x_k)$) and challenge ciphertext as there may be a correlation between them. More precisely, we observe a distributional relation between $c_0^*(\hat{s}_0, \hat{\theta}) := \hat{s}_0(\hat{\theta}_1 h^* + \hat{\theta}_2) \mod p_2$ and $c_{c^*}^*(\hat{s}, \hat{h}) := \hat{s}_0 := \hat{s}_0 \mod p_2$ with $\hat{s}_0 := \hat{z}_i s_0$ involved in $c_{c^*}^*(\hat{s}, \hat{h}_M)$ of challenge ciphertext $CT^*$. Unfortunately, a similar kind of relation is found in $\hat{b}$, viz., between $b := -t_2 \mod p_2$ and $\iota := t_2(\hat{\theta}_1 h_k + \theta_2) \mod p_2$. But that correlation does not hamper our life, if $h^* \neq h_k$. In fact, we show that $h^* \neq h_k$.

If $h^* = h_k$, then $B$ aborts the game in query-2 phase. Therefore, we only have to show that the probability of abort is negligible. By Fact 3.11, we have that the probability of abort is bounded by the advantage of an adversary in breaking DSG2 assumption. Altogether, we have that the joint distribution of all the objects simulated by $B$ is identical to that of Game2$_{-(k-1)-2}$ if $\beta = 0$ else Game2$_{2-k-1}$.

**Fact 3.11.** If for the $k^{th}$ decrypt query $(CT_k, x_k)$, $CT^* \neq CT_k$ and $h^* = h_k$, then $B$ can solve the given instance of DSG2 assumption.

**Proof of Fact 3.11.** We start with the following equalities:

$$CT^* \neq CT_k \text{ and } h^* = h_k$$

(3)

Since, $H$ is a collision resistant hash function, from the equation (3), we have

$$C_0^* \neq C_0^{(k)} \text{ and } C_{c^*}^* = C_{c^*}^{(k)}$$

(4)

From the definition of AltDecrypt, we have the following equations:

$$C_0^{(k)} |_{G_{p_3}} = 1_G \text{ and } e(g, C_0^{(k)}) = e(g^{\hat{\theta}_1 h_k + \theta_2}, C_1^{(k)})$$

(5)

From the challenge ciphertext, we have

$$C_0^* |_{G_{p_3}} = 1_G \text{ and } e(g, C_0^*) = e(g^{\hat{\theta}_1 h^* + \theta_2}, C_1^*)$$

(6)

Using the 2nd part of the equations (3), (4) (viz., $C_1^* = C_1^{(k)}$), (5) and (6), we have $e(g, C_0^*) = e(g, C_0^{(k)})$ which in turn implies that

$$C_0^* |_{G_{p_1}} = C_0^{(k)} |_{G_{p_1}}$$

(7)
Since $C_0^{(k)}|_{G_{p_3}} = 1_G$, $C_0^*|_{G_{p_3}} = 1_G$, using equation (7), we must have

$$Y_2 := (C_0^*)^{-1} \cdot C_0^{(k)} \in \mathbb{G}_{p_2}.$$ 

Since $C_0^* \neq C_0^{(k)}$, we have $Y_2 \neq 1_G$. Therefore, $\mathcal{B}$ can break the given instance of DSG2 assumption using $Y_2$.

**Lemma 3.12.** Game$_{2-k-1}$ and Game$_{2-k-2}$ are indistinguishable under the DSG2 assumption. That is, for every adversary $\mathcal{A}$, there exists a PPT algorithm $\mathcal{B}$ such that

$$|\text{Adv}_{\mathcal{A}, \text{PE}}^{2-k-1}(\kappa) - \text{Adv}_{\mathcal{A}, \text{PE}}^{2-k-2}(\kappa)| \leq \text{Adv}_{\mathcal{B}}^{\text{DSG2}}(\kappa) \text{ for } 1 \leq k \leq \nu.$$ 

**Proof.** We establish a PPT simulator $\mathcal{B}$ who receives an instance of DSG2, $(J, g, Z_1Z_2, W_2W_3, Z_3, T_3)$ with $\beta \leftarrow \{0, 1\}$ and depending on the distribution of $\beta$, it simulates either Game$_{2-k-1}$ or Game$_{2-k-2}$. The simulation is almost similar to Lemma 3.10 except the answering $k^{th}$ decrypt query. Note that in this case, we do not need the collision resistant property of $H$. We illustrate here only the $k^{th}$ alt-key: The $k^{th}$ alt-key is of either sf-type 1 or sf-type 2. $\mathcal{B}$ runs $(k_{x_k}, m_2) \leftarrow \text{Enc1}(x_k, N)$ and Pair$(x_k, y_k) \rightarrow E \in \mathbb{Z}_N^t \times \mathbb{W}_1$. Picks $i'_k \leftarrow U \mathbb{Z}_N$ and $R_3 \leftarrow U \mathbb{G}_{p_3}'$. It computes the alt-key as given below:

$$SK_{x_k}^M := (0, k_{x_k}(\alpha, r, h)E), T_3^{-1}(0, \ldots, 0), T_3^{(0, \theta_1h_k + \theta_2, \ldots, 0)}(W_2W_3)(0, i'_k, 0, \ldots, 0), R_3,$$

where $h_k := H(\psi(m_{x_k}))$.

Let $W_2W_3 = g^2_{x_k}g^w_{x_k}$. Let $g^r := T_3|_{G_{p_1}}$ and for $\beta = 1$, $g'^{t_3} := T_3|_{G_{p_1}}$. Then, $\mathcal{B}$ sets $g^v = SK_{x_k}^M|_{G_{p_2}}$, where $v := (-r, \psi + k_{x_k}(\alpha, r, h)E)$ and $\psi := (r(\theta_1h_k + \theta_2), 0, \ldots, 0)$. If $\beta = 1$ (resp. $\beta = 0$), it sets $g^0 := SK_{x_k}^M|_{G_{p_2}}$ with $\psi := (b, t, 0, \ldots, 0) \in \mathbb{Z}_N^{\nu+1},$ where $\mathcal{B}$ implicitly sets $b := -t_2 \mod p_2$ (resp. $b := 0 \mod p_2$) and $t := t_2(\theta_1h_k + \theta_2) + w_2i'_k \mod p_2$ (resp. $t := w_2i'_k \mod p_2$). Therefore, $SK_{x_k}^M$ is perfectly distributed sf-type 1 (resp. sf-type 2) alt-key if $\beta = 1$ (resp. $\beta = 0$).

**Lemma 3.13.** Game$_{2-u-2}$ and Game$_{\text{Final}}$ are indistinguishable under the DSG3 assumption. That is, for every adversary $\mathcal{A}$, there exists a PPT algorithm $\mathcal{B}$ such that

$$|\text{Adv}_{\mathcal{A}, \text{PE}}^{2-u-2}(\kappa) - \text{Adv}_{\mathcal{A}, \text{PE}}^{\text{Final}}(\kappa)| \leq \text{Adv}_{\mathcal{B}}^{\text{DSG3}}(\kappa).$$ 

**Proof.** We establish a PPT simulator $\mathcal{B}$ who receives an instance of DSG1, $(J, g, g^\alpha Y_2, g^\alpha W_2, g_2, Z_3, T_3)$ with $\beta \leftarrow \{0, 1\}$ and depending on the distribution of $\beta$, it simulates either Game$_{2-u-2}$ or Game$_{\text{Final}}$.

**Setup:** $\mathcal{B}$ chooses $\theta_1, \theta_2 \leftarrow U \mathbb{Z}_N$, $h \leftarrow U \mathbb{Z}_N^\nu$ and sets $h_M := (\theta_1, \theta_2, h)$. Let $H : \{0, 1\}^* \rightarrow \mathbb{Z}_N$ be a hash function. Then, it provides $PP := (J, g, g^h, g^f := e(g, g^\alpha Y_2), Z_3, H)$ to $\mathcal{A}$. Implicitly sets $h_M := h_M \mod p_2$. By CRT, $h_M$ is independent from $h_M \mod p_1$ and so $h_M$ is perfectly distributed.

**Query Phase-1:** It consists of the following queries in adaptive manner:

- **KeyGen**$(x)$: It is sf-type 3 key. Let $x_j$ be the $j^{th}$ query key-index. $\mathcal{B}$ runs $(k_{x_j}, m_3) \leftarrow \text{Enc1}(x_j)$. Then picks $r_j \leftarrow U \mathbb{Z}_N^{t_3}$, $\hat{r}' \leftarrow \mathbb{Z}_N$ and $R_3 \leftarrow U \mathbb{G}_{p_3}$. Finally it returns

  $$SK_{x_j} := (g^\alpha Y_2^{k_{x_j}(1, 0, 0)}, g^k_{x_j(0, r_j, h)}, g_{k_{x_j}(\hat{r}'_j, 0, 0)}, R_3.$$
If \( Y_2 = g_2^{y_2} \), \( \mathcal{B} \) implicitly sets \( \hat{\alpha}_j := y_2 + \hat{\alpha}_j \mod p_2 \) and so, \( SK_{x_j} \) is a perfectly distributed sf-type 3 key.

- **Decrypt**(CT, x): It is answered by sf-type 2 alt-key. As above, \( \mathcal{B} \) first creates sf-type 3 key \( SK_{x_j} := (x, K_x := g_{k_x(\hat{\alpha}_j, \hat{\tau}, \hat{h})}^x, g_{k_x(\hat{\alpha}_j, 0, 0)}^x \cdot R_3) \) and then using \( SK_{x_j} \), it can compute the sf-type 2 alt-key \( SK^M_{x_j} \) as follows: It computes

\[
SK^M_{x_j} := (K_0, \Psi \cdot K^E_x) \cdot g_2^{(0,i',0,...,0)} \in \mathbb{G}_p^1, \quad \text{where} \quad K_0 := g^{-\tau} \cdot R_0, \quad \tau, i' \xleftarrow{} \mathbb{Z}_N, \\
R_0 \xleftarrow{} \mathbb{G}_{p_3}, \quad \Psi := g^\psi \text{ with } \psi := (\tau(\theta_1 h + \theta_2), 0, \ldots, 0) \in \mathbb{Z}_N^m, \quad h := H(C_{cpa}) \text{ and } E \xleftarrow{} \text{Pair}(x, y).
\]

Finally it returns \( \text{Guess} := (s_1, \ldots, s_{m_2}) \xleftarrow{} \mathbb{U} \mathbb{Z}_N^{m_2} \) and \( s' := (1, s_1', \ldots, s_{m_2}') \). \( \mathcal{B} \) implicitly sets \( s := s_0 s' \mod p_1 \) and \( \hat{s} := s_0 s' \mod p_2 \). By CRT, \( s' \mod p_1 \) is independent from \( s' \mod p_2 \) and so, \( s \) and \( \hat{s} \) are perfectly distributed as required. Therefore, \( CT^* \) is perfectly distributed sf-type 1 ciphertext if \( \beta = 0 \) else sf-type 2 ciphertext.

**Query Phase-2:** It consists of the following queries in adaptive manner:

- **KeyGen** (x): It is similar to **Query Phase-1**, except \( \hat{\alpha}_j \) will be the same for all post queried keys \( x_j \). In fact, it is described here. Let \( x_j \) be the \( j \)th query key-index. \( \mathcal{B} \) runs \( (k_{x_j}, m_2) \xleftarrow{} \text{Enc1}(x_j) \). Then picks \( r_j \xleftarrow{} \mathbb{Z}_N^{m_2} \), \( \hat{\alpha}' \xleftarrow{} \mathbb{Z}_N \) and \( R_3 \xleftarrow{} \mathbb{G}_{p_3} \). Finally it returns

\[
SK_{x_j} := (g^{s_0 Y_2})^{k_{x_j}(1,0,0)} \cdot g^{k_{x_j}(0, r_j h)} \cdot g_{k_x(\hat{\alpha}'_j, 0, 0)}^x \cdot R_3.
\]

If \( Y_2 = g_2^{y_2} \), \( \mathcal{B} \) implicitly sets \( \hat{\alpha} := y_2 + \hat{\alpha}_j \mod p_2 \) and so, \( SK_{x_j} \) is a perfectly distributed sf-type 3 key.

- **Decrypt**(CT, x): Similar to **Query Phase-1**.

**Guess:** \( \mathcal{A} \) sends a guess \( b' \) to \( \mathcal{B} \). If \( b' = b \) then \( \mathcal{B} \) returns 1 else 0.

**Analysis:** All the components simulated above are perfectly distributed as required. Therefore, the joint distribution of all the objects simulated by \( \mathcal{B} \) is identical to that of Game_{2\rightarrow2} if \( \beta = 0 \) else Game_{Final}.

**Theorem 3.14.** Let \( \mathcal{P} \) be a pair encoding scheme for a predicate \( \sim \) which satisfies Conditions 2.7 and \( \sim \) is domain-transferable. Suppose \( \mathcal{P} \) has PMH security, the assumptions, DSG1, DSG2 and DSG3 hold in \( J \) and \( H \) is a collision resistant hash function, then the proposed predicate encryption scheme PE in Section 3.1 for the predicate \( \sim \) is AP-IND-CCA secure (Definition 2.6).
Proof. Similar to the proof of Theorem 3.1. The reduction of the proof is given by
\[
\text{Adv}_{\mathcal{A}}^{\text{AP-IND-CCA}}(\kappa) \leq \text{Adv}_{\mathcal{B}_1}^{\text{DSG1}}(\kappa) + (2q + 2\nu + 1)\text{Adv}_{\mathcal{B}_2}^{\text{DSG2}}(\kappa) + \nu \text{Adv}_{\mathcal{B}_3}^{\text{CRH}}(\kappa) + \text{Adv}_{\mathcal{B}_4}^{\text{DSG3}}(\kappa)
\]
where \( q \) and \( \nu \) respectively be the number of key and decrypt queries made \( \mathcal{A} \) and \( \mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4 \) are PPT algorithms whose running times are same as that of \( \mathcal{A} \).

4. Conclusion

In this paper, we have shown an efficient construction of fully CCA-secure predicate encryptions from pair encodings having almost the same cost as the CPA-secure PE of [1]. In particular, it has one extra group element in ciphertext, three extra pairing computations and one re-randomization of key in decryption as compared to the CPA-decryption of [1].

Appendix A. A brief survey of predicate encryption

Attribute-based encryption [31, 37] is a generalization of IBE [8, 18] for providing access control over the secrets in fine grained manner. Sahai and Waters [42] introduced the concept of attribute-based encryption. Since then, many ABE schemes [20, 15, 44, 25, 4, 37, 31, 14, 30] have been studied. Among them the schemes [20, 15, 44, 4] improved the efficiency, the schemes [14, 30] focused on multi-authority setting and the schemes [37, 31] improved the security level.

The security of all the aforementioned ABE schemes (except [37, 31]) was proven in selective model, a weak model where an adversary has to publish the challenge policy (in case of CP-ABE) or the set of attributes (in case of KP-ABE) before seeing the public parameters. In contrast, the adversary has a flexibility to choose the challenge policy or the set of attributes in challenge phase of adaptive security model. Lewko et al. [31] first took a big step forward in the construction of adaptively secure ABE schemes for monotone access structures in the standard model. The authors show that how to utilize the dual system methodology [45] of Waters in the area of ABE. For utilizing this methodology, the authors [31] used DSG assumptions in composite order bilinear groups. Later, Okamoto and Takashima [37] reached to the adaptively secure predicate encryption schemes with general relations in the prime order bilinear groups. Their constructions are based on the concept of dual pairing vector spaces [36]. The authors [37] used decisional subspace assumptions to abstract the dual system methodology. These subspace assumptions are reduced from the decisional linear (DLIN) assumption.

Similar to the traditional ABE, there are many other encryptions available in the literature. Some of them are ABE for circuits, (doubly-)spatial encryption ((D-)SE), functional encryption for regular languages, (hierarchical) inner-product encryption ((H)IPE), etc. These encryptions are subsumed under a larger class of encryptions, called predicate encryption. These encryptions are briefly described below.

- (ABE for Circuits.) Traditional ABE [26, 4, 46] provides access control over data using polynomial-size boolean formulas or equivalently circuits with fanout 1. Gorbufov, Vaikuntanathan and Wee [24] and Garge et al. [22] first independently proposed ABE constructions for circuits of arbitrary fanout.
The construction of [22] is based on multi-linear maps, whereas the construction of [24] is based on lattice. All the previous ABE schemes [24, 22] for general circuits were proven selectively secure. Then, Attrapadung [2] proposed first adaptively secure constructions for general circuits.

- (Spatial Encryption.) Spatial encryption was introduced by Boneh and Hamburg [10] as a special instance of generalized identity-based encryption (GIBE). The GIBE [10] captures many encryptions, e.g., traditional IBE, broadcast IBE, HIBE and forward-secure system, etc. One of the main building blocks used for GIBE is the spatial encryption [10, 51, 32], where a key-index is represented by affine subspace of an affine space and data-index is a point of the affine space. The access is granted if the data-index is a member of the key-index. Boneh and Hamburg [10] showed how to obtain different flavor of IBE, e.g., HIBE, inclusive IBE, co-inclusive IBE, broadcast IBE, product scheme, multiple authority scheme, forward-secure scheme, etc., from spatial encryption. Later, Hamburg [27] extended the notion of spatial encryption to doubly-spatial encryption (DSE) from which expressive predicate encryptions, e.g., IPE, ABE, etc. can be derived. In doubly-spatial encryption [27, 16, 17], key-index and data-index are represented by affine subspaces. The key-index satisfies the associated data-index if and only if their intersection is not void.

- (Inner Product Encryption.) Katz, Sahai and Waters [28] introduced the predicate encryption with hidden index for inner product relation. In this PE, the access control over a data is defined through the orthogonality (zero inner-product) of key-index and data-index. This form of PE is called zero IPE or simply IPE [40, 39, 31, 38]. Its dual form, where the relation is defined through non-orthogonality (non-zero inner-product) is called non-zero IPE [3, 38]. The IPE schemes are mainly constructed for data-index hiding, but some of them [3, 38] are available to handle only payload hiding. The IPE schemes are available in either hierarchical or non-hierarchical style.

- (Functional Encryption for Regular Languages.) All the aforementioned schemes can provide at most bounded access. Waters [47] first moved to unbounded access control system using regular languages as policies. In this system, key-indices are labeled with regular languages over an alphabet and data-indices are associated with strings over the same alphabet. This system is called key-policy functional encryption (KP-FE) for regular languages. The construction was proven to be selectively CPA-secure under decisional $\ell$-Expanded BDHE assumption. Recently, Attrapadung [1] proposed first adaptively CPA-secure KP-FE and CP-FE for regular languages.

References

[1] N. Attrapadung, Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more, In EUROCRYPT, Lecture Notes in Comput. Sci., volume 8441, Springer, 2014, 557–577.

[2] N. Attrapadung, Fully secure and succinct attribute based encryption for circuits from multilinear maps, Cryptology ePrint Archive, Report 2014/772, 2014, http://eprint.iacr.org/.

[3] N. Attrapadung and B. Libert, Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation, In PKC, Lecture Notes in Comput. Sci., volume 6056, Springer, 2010, 384–402.
[4] N. Attrapadung, B. Libert and E. Panafieu, Expressive key-policy attribute-based encryption with constant-size ciphertexts, In PKC, Lecture Notes in Comput. Sci., volume 6571, Springer, 2011, 90–108.

[5] N. Attrapadung and S. Yamada, Duality in ABE: Converting attribute based encryption for dual predicate and dual policy via computational encodings, In CT-RSA, Lecture Notes in Comput. Sci., volume 9048, Springer, 2015, 87–105.

[6] J. Blömer and G. Liske, Construction of fully cca-secure predicate encryptions from pair encoding schemes, In CT-RSA, Lecture Notes in Comput. Sci., volume 9610, Springer, 2016, 431–447.

[7] D. Boneh, R. Canetti, S. Halevi and J. Katz, Chosen-ciphertext security from identity-based encryption, Journal of SIAM, 36 (2007), 1301–1328.

[8] D. Boneh and M. Franklin, Identity-based encryption from the weil pairing, In CRYPTO, Lecture Notes in Comput. Sci., volume 2139, Springer, 2001, 213–229.

[9] D. Boneh, E. Goh, and K. Nissim, Evaluating 2-dnf formulas on ciphertexts, In TCC, Lecture Notes in Comput. Sci., volume 3378, Springer, 2005, 325–341.

[10] D. Boneh and M. Hamburg, Generalized identity-based and broadcast encryption schemes, In ASIACRYPT, Lecture Notes in Comput. Sci., volume 5350, Springer, 2008, 455–470.

[11] D. Boneh and J. Katz, Improved efficiency for CCA-secure cryptosystems built using identity-based encryption, In CT-RSA, Lecture Notes in Comput. Sci., volume 3376, Springer, 2005, 87–103.

[12] X. Boyen, Q. Mei and B. Waters Direct chosen ciphertext security from identity-based techniques, In ACM Conference on Computer and Communications Security, ACM, New York, 2005, 320–329.

[13] R. Canetti, S. Halevi and J. Katz, Chosen-ciphertext security from identity-based encryption, In EUROCRYPT, Lecture Notes in Comput. Sci., volume 3027, Springer, 2004, 207–222.

[14] M. Chase, Multi-authority attribute based encryption, In TCC, Lecture Notes in Comput. Sci., volume 4392, Springer, 2007, 515–534.

[15] C. Chen, Z. Zhang and D. Feng, Efficient ciphertext policy attribute-based encryption with constant-size ciphertext and constant computation-cost, In PROVSEC, Lecture Notes in Comput. Sci., volume 6980, Springer, 2011, 84–101.

[16] C. Chen, Z. Zhang and D. Feng, Fully secure doubly-spatial encryption under simple assumptions, In PROVSEC, Lecture Notes in Comput. Sci., volume 7496, Springer, 2012, 253–263.

[17] J. Chen and H. Wee, Doubly spatial encryption from DBDH, Theoret. Comput. Sci., 543 (2014), 79–89.

[18] C. Cocks, An identity based encryption scheme based on quadratic residues, In Cryptography and Coding, Lecture Notes in Comput. Sci., volume 2260, Springer, 2001, 360–363.

[19] W. Diffie and M. Hellman, New directions in cryptography, IEEE Transactions on Information Theory, 22 (1976), 644–654.

[20] K. Emura, A. Miyaji, A. Nomura, K. Omote and M. Soshi, A ciphertext-policy attribute-based encryption scheme with constant ciphertext length, In ISPEC, Lecture Notes in Comput. Sci., volume 5451, Springer, 2009, 13–23.

[21] E. Fujisaki and T. Okamoto, Secure integration of asymmetric and symmetric encryption schemes, In CRYPTO, Lecture Notes in Comput. Sci., volume 1090, Springer, 1996, 109–118.

[22] S. Garg, C. Gentry, S. Halevi, A. Sahai and B. Waters, Attribute-based encryption for circuits from multilinear maps, In CRYPTO, Lecture Notes in Comput. Sci., volume 8043, Springer, 2013, 479–499.

[23] C. Gentry and A. Silverberg, Hierarchical ID-based cryptography, In ASIACRYPT, Lecture Notes in Comput. Sci., volume 2501, Springer, 2002, 548–566.

[24] S. Gorbunov, V. Vaikuntanathan and H. Wee, Attribute-based encryption for circuits, In STOC’13–Proceedings of the 2013 ACM Symposium on Theory of Computing, ACM, 2013, 545–554.

[25] V. Goyal, A. Jain, O. Pandey and A. Sahai, Bounded ciphertext policy attribute based encryption, In Automata, Languages and Programming. Part II, Lecture Notes in Comput. Sci., volume 5126, Springer, 2008, 579–591.
[26] V. Goyal, O. Pandey, A. Sahai and B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, In ACM Conference on Computer and Communications Security, ACM, 2006, 89–98.

[27] M. Hamburg, Spatial encryption, Cryptology ePrint Archive, Report 2011/389, 2011, http://eprint.iacr.org/.

[28] J. Katz, A. Sahai and B. Waters, Predicate encryption supporting disjunctions, polynomial equations, and inner products, In EUROCRYPT, Lecture Notes in Comput. Sci., volume 4965, Springer, 2008, 146–162.

[29] A. Lewko and B. Waters, New techniques for dual system encryption and fully secure HIBE with short ciphertexts, In TCC, of Lecture Notes in Comput. Sci., volume 5978, Springer, 2010, 455–479.

[30] A. Lewko and B. Waters, Decentralizing attribute-based encryption, In EUROCRYPT, Lecture Notes in Comput. Sci., volume 6632, Springer, 2011, 568–588.

[31] A. Lewko, T. Okamoto, A. Sahai, K. Takashima and B. Waters, Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption, In EUROCRYPT, Lecture Notes in Comput. Sci., volume 6110, Springer, 2010, 62–91.

[32] D. Moriyama and H. Doi, A fully secure spatial encryption scheme, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 94 (2011), 28–35.

[33] M. Nandi and T. Pandit, Generic conversions from CPA to CCA secure functional encryption, Cryptology ePrint Archive, Report 2015/457, 2015, http://eprint.iacr.org/.

[34] M. Nandi and T. Pandit, Verifiability-based conversion from CPA to CCA-secure predicate encryption, Appl. Algebra Engrg. Comm. Comput., 29 (2018), 77–102.

[35] M. Nandi and T. Pandit, Delegation-based conversion from CPA to CCA-secure predicate encryption, International Journal of Applied Cryptography, 4 (2020), 16–35.

[36] T. Okamoto and K. Takashima, Hierarchical predicate encryption for inner-products, In ASIACRYPT, Lecture Notes in Comput. Sci., volume 5912, Springer, 2009, 214–231.

[37] T. Okamoto and K. Takashima, Fully secure functional encryption with general relations from the decisional linear assumption, In CRYPTO, Lecture Notes in Comput. Sci., volume 6223, Springer, 2010, 191–208.

[38] T. Okamoto and K. Takashima, Achieving short ciphertexts or short secret-keys for adaptively secure general inner-product encryption, In Cryptology and Network Security, Lecture Notes in Comput. Sci., volume 7092, Springer, 2011, 138–159.

[39] T. Okamoto and K. Takashima, Adaptively attribute-hiding (hierarchical) inner product encryption, In EUROCRYPT, Lecture Notes in Comput. Sci., volume 7237, Springer, 2012, 591–608.

[40] T. Okamoto and K. Takashima, Fully secure unbounded inner-product and attribute-based encryption, In ASIACRYPT, Lecture Notes in Comput. Sci., volume 7658, Springer, 2012, 349–366.

[41] R. Rivest, A. Shamir and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Comm. ACM, 21 (1978), 120–126.

[42] A. Sahai and B. Waters, Fuzzy identity-based encryption, In EUROCRYPT, Lecture Notes in Comput. Sci., volume 3494, Springer, 2005, 457–473.

[43] A. Shamir, Identity-based cryptosystems and signature schemes, In CRYPTO, Lecture Notes in Comput. Sci., volume 196, Springer, 1984, 47–53.

[44] B. Waters, Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization, In PKC, Lecture Notes in Comput. Sci., volume 6571, Springer, 2011, 53–70.

[45] B. Waters Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions, In CRYPTO, Lecture Notes in Comput. Sci., volume 5677, Springer, 2009, 619–636.

[46] B. Waters, Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization, In PKC, Lecture Notes in Comput. Sci., volume 6571, Springer, 2011, 53–70.

[47] B. Waters, Functional encryption for regular languages, In CRYPTO, Lecture Notes in Comput. Sci., volume 7417, Springer, 2012, 218–235.
[48] H. Wee, Dual system encryption via predicate encodings, In *TCC*, Lecture Notes in Comput. Sci., volume 8349, Springer, 2014, 616–637.

[49] S. Yamada, N. Attrapadung, G. Hanaoka and N. Kunihiro, Generic constructions for chosen-ciphertext secure attribute based encryption, In *PKC*, Lecture Notes in Comput. Sci., volume 6571, Springer, 2011, 71–89.

[50] S. Yamada, N. Attrapadung, B. Santoso, J. C. N. Schuldt, Goichiro Hanaoka and Noboru Kunihiro, Verifiable predicate encryption and applications to CCA security and anonymous predicate authentication, In *PKC*, Lecture Notes in Comput. Sci., volume 7293, Springer, 2012, 243–261.

[51] M. Zhou and Z. Cao, Spatial encryption under simpler assumption, In *PROVSEC*, Lecture Notes in Comput. Sci., volume 5848, Springer, 2009, 19–31.

Received for publication January 2020.

E-mail address: mridul@isical.ac.in
E-mail address: tapasgmmath@gmail.com